Temporal deep learning for financial time series

by

Muhammed Kehinde Olorunnimbe

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The widespread usage of machine learning in different mainstream contexts has made deep learning the solution of choice in various domains, including finance. However, the real-world application of deep learning in the stock market is still emerging. When practitioners are applying machine learning to the stock market, there are three primary stages, namely (1) data/feature engineering, (2) price forecast, and (3) investment decision and strategy. Considerations across all three stages include the unique characteristics of financial time series data, the need to understand model predictions, and ease of use. Historically, deep learning is considered to not excel under these considerations, hence it is common to see classical statistical and traditional machine learning methods in use by market practitioners. In this Ph.D. thesis, we focus on advancing this area of research by combining the state-of-the-art in both financial statistics and deep learning.

In this thesis, we put financial-specific data attributes into consideration to produce novel algorithms based on the *temporal Transformer* deep learning architecture, a state-of-the-art deep learning approach that draws dependencies between data sequences using a mechanism called *attention*. By introducing temporal Transformers focused on the financial domain, we illustrate its predictive use case for financial time series. We also incorporate discussions on explainable AI (XAI), to mitigate the black-box nature often associated with such deep learning algorithms.

We address the first two stages by introducing similarity embedded temporal Transformer (SeTT) and run-similarity embedded temporal Transformer (r-SeTT) algorithms that combine temporal Transformer architecture with time series forecasting and statistical principles. We employ similarity vectors generated from historical trends across different financial instruments that are used to adjust the weight of the temporal Transformer model during the training process. This approach takes advantage of the conditional heteroscedasticity in financial time series, by using the historical volatility in combination with the attention mechanism in a temporal Transformer deep neural network architecture. Our experimentation shows that by focusing on the historical windows that are most similar to the current window in the attention-tuning process, we outperform both classical financial models and the baseline temporal Transformer model in terms of predictive performance.

To effectively utilize an extended history of financial time series data, we further develop an ensemble algorithm called windowing ensemble of temporal Transformers (WETT). Our ensemble algorithm leverages a combination of base models generated from sliding windows of historical timeframes, with additional weight initialization diversification options for a complete experimentation regime. By decomposing the constituent time series of the

extended timeframe, we optimize the utilization of the series for financial deep learning. This simplifies the training process while achieving better performance, particularly when accounting for the non-constant variance of financial time series.

To address the last stage of applying complex deep learning architectures to the financial market, we examined advancements in XAI and its application in facilitating investment decision-making processes. Our discussion centers on incorporating XAI into our model development pipeline, through the use of surrogate models. This step aims not only to augment our comprehension of the temporal Transformer model but also to improve the model's predictive capabilities by assessing the effectiveness of the input features.

Acknowledgements

All praise and gratitude are due to God, Almighty. I dedicate this achievement to my beloved wife, Jem; without your unwavering love and support throughout this challenging journey, I doubt I could have persevered. I love you.

My deepest appreciation goes to my supervisor, Dr. Herna L. Viktor. From the initial trust placed in me to join your research lab during my Master's degree to your guidance through my Doctorate, despite my numerous missteps, I am profoundly grateful. Thank you! I also extend my heartfelt thanks to Dr. Eric Paquet, who introduced me to Transformer architecture over three years ago, steering my research in a new and rewarding direction after a challenging period.

I am deeply grateful to my parents for their endless sacrifices and unconditional love growing up. Additionally, I extend my heartfelt thanks to my siblings and friends – Aminat, Dr. Jide, Mustapha, Ismail, Nafisat, Fatai, Fuad, Dr. Okhaide, Dr. Bisi, Neha, Kavya, Vanessa, Akeem, and others – for their encouragement and emotional support throughout this journey. Your constant support and motivation were invaluable.

Lastly, I am grateful for the support from the University of Ottawa and Dr. Herna Viktor, whose accommodations and financial assistance were crucial in helping me towards this significant milestone.

Dedication

To the continuous pursuit of self-improvement, selflessness, and the striving towards a greater purpose.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

H_0 null hypothesis 68

AI artificial intelligence 1, 32, 104

ANN artificial neural networks 19, 20, 22–24, 26, 27, 32, 38

API application programming interface 12, 14, 50

ARIMA autoregressive integrated moving average 79, 84, 95

CAE convoluted autoencoder 27, 44, 55

CAGR compound annual growth rate 33, 34

CBOE Chicago Board Options Exchange 33, 76

CM Cramér–von Mises 68

CNN convolutional neural network 18, 26, 27, 39, 44–46, 48, 49, 53, 55, 57, 89

DBN deep belief networks 19

DeepAR probabilistic forecasting with autoregressive recurrent networks 79, 84, 95, 100

DJIA Dow Jones industrial average 74, 75, 85, 101

DQN deep Q-network 44

DRL deep reinforcement learning 28, 29, 39, 43, 44, 46–48, 50, 53, 56, 120

EAIML Ethical AI & machine learning 104

EDA exploratory data analysis 105, 108, 111, 113

EGL explanation-guided learning viii, xiv, 105, 106, 109–114, 118–120

ES Epps–Singleton 68, 69, 104, 108

FAIR Findability, Accessibility, Interoperability and Reusability 75

FFNN feed-forward neural network 22, 24, 39, 44, 46–50, 55, 57, 62

FIX financial information exchange 12

GARCH generalized autoregressive conditional heteroscedasticity 79, 84, 95

GPU graphics processing unit 116

GRN gated residual network 71

GRU gated recurrent unit 26, 60

hd Hamming distance 69, 104, 108

HMM Hidden Markov model 49

IID independent and identically distributed 4, 32, 37, 65, 68, 69

KLD Kullback-Leibler divergence 45

KS Kolmogorov–Smirnov 68

LIME local interpretable model-agnostic explanations 105

LLM large language models 62, 116

LOB limit order book 45, 55

LSTM long short-term memory 26, 39, 43–50, 53–57, 60, 63, 71, 104

MACD moving average convergence/divergence 44

MAE mean absolute error 36, 43, 73

MAPE mean absolute percentage error 36, 43

MASE mean absolute scaled error 7, 73, 80, 82, 112

MDD maximum drawdown 33, 34

MDP Markov decision process 28, 46

MSE mean square error 36, 42

NHiTS neural hierarchical interpolation for time series forecasting 79, 84, 95

NLP natural language processing 2, 60, 62, 63

NPUs neural processing Units 96

OHLC open-high-low-close 18

PCA principal component analysis 48, 57

PReLU parametric ReLU 22

QRNN quantile regression neural network 92

r-SeTT run-similarity embedded temporal Transformer ii, xi, 4, 67, 69, 78, 80, 84, 85, 87, 89, 93, 95, 96, 98, 104, 107–113, 117, 118

RBM restricted Boltzmann machine 40

ReLU rectified linear unit 22

ResNet residual network 44, 47, 53, 56, 71

RL reinforcement learning 27, 28, 39

RMSE root mean square error 73

RNN recurrent neural network 25, 26, 39, 45, 49, 57, 60, 79, 80, 84, 95, 100

RoR rate of returns 34

SeTT similarity embedded temporal Transformer ii, xi, 4, 64–68, 78, 80–85, 87, 89, 93, 95, 96, 98, 104, 107, 108, 117, 118

SimFin SimFin Analytics GmbH 74, 75

SVM support vector machines 46, 55

TFT temporal fusion Transformer xi, 63, 64, 79–85, 87, 89, 93, 95–98, 100, 104, 107, 108

TQDN trading deep Q-Network 44

TS-MULE time series multivariate and univariate local explanations 105, 108, 109, 111, 118

VaR value at risk 34, 35, 49, 57

VIX volatility index 33, 76, 108

VWAP volume-weighted average price 17

WETT windowing ensemble of temporal Transformers ii, xi, 86–88, 90, 93, 94, 96–100, 117, 118, 120

WRDS Wharton research data services 50

XAI explainable AI ii, iii, 6, 7, 104, 107, 113–115, 118–120

Chapter 1

Introduction

1.1 Background

Technology has long been a great enabler for various financial innovations (Seese et al., 2008). This could be either by way of enhancing traditional ways of conducting financial processing such as risk management or by unlocking new potentials that would not be previously possible such as algorithmic trading. It is no wonder that the world of finance embraces artificial intelligence (AI) and machine learning as a different way of getting more value from the vast amount of available data (Prado, 2018).

As these machine learning use cases continue to grow, the popularity of its different techniques has also gone through numerous phases. In recent years, deep learning approaches have outperformed classical machine learning methods in a variety of tasks, including financial applications. In (Insights, 2019), Deloitte surveyed over 200 U.S. financial service executives to determine their use for AI and how it impacts their business. A total of 70% of respondents indicated that they use machine learning in general, while 52% indicated that they use deep learning. The common use cases for deep learning amongst these responses are reading claims documents for triage, providing data analytics to users through intuitive dashboards, and developing innovative trading and investment strategies in the financial market. Transformer-based models have emerged as state-of-the-art in deep learning approaches. The development of Transformer-based large language models (LLMs) like BERT and InstructGPT has significantly advanced artificial intelligence, increasing its presence in public discussions. This evolution has led to Generative AI, opening up new practical applications with substantial economic impacts (Luk, 2023).

Transformer models are increasingly applied to various machine learning scenarios in the financial domain including financial market forecasting, as they have demonstrated the potential to outperform traditional statistical methods (Lim, Arik, et al., 2021). When applying machine learning in the financial market, there are three stages the overall process follows (Chan, 2016; Chan, 2020; S. Sun et al., 2023), namely:

1. Data/feature engineering: This stage is related to understanding the characteristics of financial data, and it involves data procurement, cleansing, analysis and transformation. Typically, this process is time-consuming and requires a significant amount of domain expertise.

2. Price forecast involves applying machine learning models on data to make predictions. The output of this stage is one or more evaluation metrics to make decisions on. It is recommended not to use this output as a trade signal, but instead, feed it into a more elaborate investment strategy with domain-specific considerations.

3. Investment strategy: This involves leveraging predictions from machine learning models for financial decision-making. Specifically, the models' outputs are used to guide a deliberate plan of action aiming to achieve targeted financial goals. This strategy is systematically evaluated based on key financial metrics, including returns and volatility, and is rigorously backtested on historical data.

Our goal in this thesis is to advance the state of deep learning research in the financial domain across these three stages. Specifically, our contributions significantly enhance the first two through the application of the Transformer architecture adapted for temporal financial data. This adaptation also improves the explainability of the model, laying essential groundwork for the third stage. However, the full development of robust investment strategies based on these enhancements is not done in this thesis and remains an area for future exploration.

The introduction of the Transformer architecture marked a pivotal moment for deep learning with sequential data like financial time series (Vaswani et al., 2017), and there have been a number of works related to its application in the financial market. However, early work primarily uses a Transformer embedding for natural language processing (NLP) preprocessing, as input to the prediction task (Davchev et al., 2020). A primary limitation of these approaches is that they are not taking full advantage of the Transformer architecture, and its attention mechanism. Rather, they only use its transfer learning capability in a NLP based multimodal learning approach that combines trading data with alternative text-based data. By not directly using the Transformer model, the opportunity is also

lost to take advantage of one of its integral contributions to model explainability via its attention network. Another major issue is because, in its original form, the Transformer architecture is insensitive to the locality of temporal data i.e., all elements within the sequence are considered equidistant to each other (Shiyang Li et al., 2019). In recent times, however, work has been done to improve the state-of-the-art in temporal Transformer models (Shiyang Li et al., 2019; Lim, Arik, et al., 2021; Yoo et al., 2021), but these works do not consider financial market-specific data characteristics in the model architecture.

1.2 Research contributions

In this work, we introduce novel temporal Transformer architectures, with financial-specific considerations, to address research challenges across the three application stages introduced above. Notably, our contributions are demonstrated through five peer-reviewed papers:

1. Kenniy Olorunnimbe and Herna L. Viktor (June 2022a). "Deep learning in the stock market—a systematic survey of practice, backtesting, and applications". In: *Artificial Intelligence Review* 56.3, pp. 2057–2109. DOI: 10.1007/s10462-022-10226-0

2. Kenniy Olorunnimbe and Herna L. Viktor (October 2022b). "Similarity embedded temporal Transformers: Enhancing stock predictions with historically similar trends". In: *26th International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pp. 388–398. DOI: 10.1007/978-3-031-16564-1_37

3. Kenniy Olorunnimbe and Herna L. Viktor (March 2024b). "Ensemble of temporal Transformers for financial time series". In: *Journal of Intelligent Information Systems*. DOI: 10.1007/s10844-024-00851-2

4. Kenniy Olorunnimbe and Herna L. Viktor (April 2024c). "Towards efficient similarity embedded temporal Transformers via extended timeframe analysis". In: *Complex & Intelligent Systems*. DOI: 10.1007/s40747-024-01400-8

5. Kenniy Olorunnimbe and Herna L. Viktor (June 2024a). "Enhancing temporal Transformers for financial time series via local surrogate interpretability". Accepted for presentation in 27th International Symposium on Methodologies for Intelligent Systems (ISMIS)

We set out to tackle the following key research challenges:

1. Consideration for the volatility, i.e., the variance in asset prices over a given time-frame, and randomness properties of financial time series data in the state-of-the-art deep learning research. The objective is to design novel temporal Transformer models with specific consideration for financial data characteristics.

2. With the expanding volume of historical financial time series, it becomes crucial to effectively utilize this data. The objective is to harness this growing data pool in a manner that maximizes efficiency without compromising on the performance of the model.

3. Design further enhancements to the architecture of temporal Transformer models for financial time series. We set out to improve the generalization properties and boost the predictive performance of temporal deep learning for financial data, across financial time series of different lengths and market conditions.

4. Assess the feature importance and relevance of temporal Transformers, contributing to the body of research involving explainability in financial deep learning. The objective is to refine feature selection and boost model performance by leveraging feature insights from the model. This clarity offers crucial guidance for model updates and eventually, decision-making. This approach brings us nearer to creating an investment strategy using temporal Transformers.

Our contributions address these challenges, formulated as research problems, as listed below.

1.2.1 Research problem 1: Similarity embedded temporal Transformers

The Transformer architecture assumes homoscedasticity, i.e. constant mean and variance, in the attention search space (Vaswani et al., 2017). This architecture is not known to be sensitive to *local context* in time series data. Local context refers to the tokens or feature vectors immediately surrounding a given input sequence from the historical data sequence. Specifically, the traditional dot-product self-attention mechanism used in the original representation of the Transformer model does not take into account the local context of the input vector, leading to anomalies and optimization issues (Shiyang Li et al., 2019). This may also make the model less effective in capturing the complex relationships that exist in financial time series data, which are often not independent and identically distributed (IID) and have non-constant variance.

As a first contribution, we introduce novel similarity embedded temporal Transformer (SeTT) architectures, which address the issue of heteroscedasticity in financial data by incorporating similarity vectors into the temporal Transformer (Olorunnimbe and Viktor, 2022b). The SeTT & r-SeTT models employ proven statistical techniques to address the non-constant variance of financial data and enhance the performance of a baseline temporal Transformer model. The outputs of these models are multi-horizon quantile forecasts across the forecast horizon. The architecture also factors in apriori known inputs, such as event date, suggesting that the effect of known corporate events such as product launches or financial earnings calls are reflected in the model's performance (Lim, Arik, et al., 2021).

This research work and experimental results are reported in Chapter 3.

1.2.2 Research problems 1 and 2: Finding optimal window in financial time series via extended timeframe analysis

To find the optimal hyperparameters for our contribution in Section 1.2.1, we conduct extensive preliminary experimentation to assess the efficacy of the models via hyperparameter optimization and extended timeframe analysis. The preliminary experimentation involves an extensive evaluation process of over 15,000 optimization trials across multiple years, timeframes, and market conditions, therefore providing a robust and reliable tool for financial analysis. We found that despite having a sufficiently extensive time series, temporal Transformer models fail to fully utilize the entirety of the time series data most of the time. This highlights the critical need for enhancing model capabilities to effectively harness and learn from the depth of historical data available, particularly in financial markets where leveraging insights from past economic cycles is essential for accurate forecasting.

We specifically address the need for improved utilization of extensive time series data, presenting analysis and experimental results reported in Chapter 3. We further use the insights from these results to introduce an ensemble approach for temporal Transformers, as reported in Chapter 4.

1.2.3 Research problem 3: Efficient utilization of extended financial time series in an ensemble of temporal Transformers

In the analysis presented in Section 1.2.2, we determined that the optimal timeframe for analyzing a historical time series spanning more than 13 years falls within shorter periods ranging from 1 to 3 years. As we strive towards optimizing model performance to achieve

the highest possible level of precision, one approach that has been shown to improve the accuracy of machine learning models is ensemble techniques. Ensemble learning involves combining multiple models, referred to as *base models*, to enhance predictive accuracy and robustness by leveraging their collective insights (Russell and Norvig, 2021). This contribution introduces a novel method that employs ensemble learning with temporal Transformer models, leveraging shorter timeframes to develop the base models. As combination methods, we use techniques such as averaging, voting, or applying a machine learning algorithm to determine the optimal way of combining these base models' predictions.

This is primarily done to reduce the bias and variance in the individual base learners (Russell and Norvig, 2021), thereby achieving a bias-variance tradeoff that improves the performance of the overall ensemble model allowing for better generalization to unseen data. This is advantageous for complex processes such as deep learning, where individual models are susceptible to overfitting the training data (Ganaie et al., 2022; Russell and Norvig, 2021). In ensemble learning, a promising approach for combining the predictions of the base models for regression problems such as this is to take the average of the individual models' outputs (Russell and Norvig, 2021). We also explored a *meta-learning*, or stacking approach to determine the best way to combine base model outputs by learning their optimal weights. This involves training an additional model, the meta-learner, to effectively blend the base models' predictions by learning the importance of each model's output.

This research work and experimental results are reported in Chapter 4.

1.2.4 Research problem 4: Enhancing temporal Transformers for financial time series via local surrogate interpretability

Despite the previously discussed innovations of the Transformer architecture, the models remain largely black-box models, i.e., their internal decision-making processes are not readily interpretable. The models' attention mechanism, while facilitating a degree of relationship mapping among encoded inputs, still presents a technical narrative tailored for a specialized audience, often out of the usual scope of stakeholders such as investors and market practitioners. This issue might be less pronounced during phases of consistent positive returns, but it becomes significantly more critical when these models underperform or when their outputs deviate from market expectations. The capability to intuitively interpret these models is not only crucial for correcting erroneous assumptions during experimentation but also plays a role in enhancing model performance. The need for comprehensible model interpretation and the ability to refine based on such insights falls

under the broad domain of XAI (Gunning and Aha, 2019).

In Section 1.1, we identified the three stages involved in the application of machine learning applications in the financial market, namely data/feature engineering, machine learning, and investment strategy. We show improvements to the state-of-the-art in the form of temporal Transformers based on multi-horizon quantile forecast, and we propose approaches to further increase said improvements. This contribution serves as a step towards closing the loop on deep learning applications to finance as it is focused on making the application of Transformer architectures in financial forecasting clearer and more understandable while also improving the predictive performance.

The experimental results are reported in Chapter 5.

1.3 Research scope

Thesis Concentration - The concentration of this work is on the application of deep learning in the financial market. We focus on designing deep learning architectures aimed at improving the performance of financial time series data, in terms of regression forecast.

Experiments - Our experiments are performed using real historical financial time-series data (SimFin, 2021). Implicit in the data is the volatility and heteroscedasticity of the financial market, and our research makes use of that to design novel temporal deep learning architectures.

Evaluations - For the predictive performance of our models, we will measure and compare the prediction loss during multiple testing periods. We used mean absolute scaled error (MASE) as the evaluation metric because it is resistant to outliers inherent in financial time series, and it independently scales the predictions with the actual target (R. J. Hyndman and Koehler, 2006). We also used quantile loss metrics because the models are trained with quantile loss functions.

1.4 Research methodology

We followed the iterative and incremental design science research (DSR) methodology (Brocke et al., 2020), consisting of six stages namely:

1. Problem identification and motivation: We surveyed existing work to determine the current state and limitations of works that focus on the application of deep learning in the financial market. After our survey, we identified gaps that need to be addressed.

2. Define the objectives for a solution: We formulated a hypothesis to combine traditional statistical functions with the state-of-the-art in deep neural networks, as existing work seldom considers domain-specific characteristics of financial time series. We also explore the use of XAI in order to offer unique insights into the domain.

3. Design and development: We collected relevant financial data to use in our research, as well as the design and implementation of our solution.

4. Demonstration: We conducted multiple experiments to obtain the optimal setup and validation for our four contributions.

5. Evaluation: A pivotal step that involves evaluating the data generated by our experimentation and validating our assumptions. We also validated that our results are not random by running statistical hypothesis tests.

6. Communication: We reported on observations and results of experiments.

1.5 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 provides the survey work that we have done about deep learning in the financial market. In Chapter 3, we introduce our first set of algorithms. We provide the details of the architectures and conduct an extensive experimental evaluation using real-world data to validate our hypothesis. Building on the first algorithms, in Chapters 4, we introduce an ensemble approach for temporal Transformers over extended timeframes, showcasing superior performance against benchmark models and single instances of the same base learners within the ensemble. In Chapter 5 we explore enhancing the explainability of the temporal Transformer model via surrogate models. We demonstrate advancements in both comprehending the model's outputs and enhancing its predictive performance. Chapter 6 concludes this thesis and highlights our future work.

Chapter 2

Deep learning with financial time series: An overview

This chapter is based on our published paper, namely:

- *Deep Learning in the Stock Market – A Systematic Survey of Practice, Backtesting, and Applications* published in Springer Artificial Intelligence Review (Olorunnimbe and Viktor, 2022) (Olorunnimbe and Viktor, 2022a).
- In this paper, we explored various scenarios employing deep learning in the stock market. We focused on research papers involving backtesting as we considered that to be criteria for real-world use cases. We recognized that domain-specific metrics such as "returns" and "volatility" appear most important for accurately representing model performance across seven distinct specializations. We also observed that while there are improvements in reproducibility, substantial work still remains regarding model explainability.

2.1 Overview

Not unlike other machine learning applications, data represents a crucial component of the stock market learning process (Prado, 2018). Understanding the different forms of data that are employed to utilize deep learning for the stock market substantially contributes to enabling proper identification of our data requirements in accordance with the task in

question. In this chapter, we consider the different characteristics, types, and representations of data that are relevant to mining stock market data using deep learning. Notably, as will become evident, some of these data forms are quite specific to stock market data.

2.2 Time series data in finance

2.2.1 Data characteristics

Source: Although trading venues such as stock exchanges are often perceived as the main source of stock market data, in recent years, other data sources, including news articles and social media, have been explored as data sources for machine learning processes (Adosoglou et al., 2020; Day and Lee, 2016; Haibe-Kains et al., 2020; S. Y. Yang et al., 2018). There is a direct correlation between data source and data type, as Section 2.2.2 demonstrates. Data source also largely depends on the intended type of analytics. If the goal is a simple regression task using purely historical market data, then the primary or only source could be trading data from the trading venue. For more complicated tasks, such as studying the effect of user sentiments on stock movement, it is common to combine trading data with data obtained from social media services or comments on relevant news articles. Irrespective of complications associated with the task at hand, it is rare to not use the trading venue as a source because literal data is always integral. Although several of the studies considered do not incorporate trading data—e.g., (Bao and Liu, 2019; Ferguson and Green, 2018)—these are generally theoretical studies that utilize simulated data.

Frequency: Data frequency concerns the number of data points within a specific unit of time (Prado, 2018). What any particular data point captures can be reported in different ways, from being represented as an aggregate (e.g., min, max, average) to using actual values. Data granularity can range from a daily snapshot (typically the closing value for trading data) to a fraction of a second for high-frequency market data. A more established representation of stock market data as *bars* (Section 2.2.3) refers to presenting multiple data points as an understandable aggregate of the highlights within that time interval.

For non-traditional data sources, such as news or social media, it is quite common to combine and summarize multiple individual items within the same time interval. For example, (Day and Lee, 2016) uses multiple daily news headlines as part of the training data. Elsewhere, using a sentence encoder (Conneau et al., 2017) generates equal-length vectors from differently sized sets of words representing different sentences. The literature

reviewed commonly uses a snapshot or aggregated data to summarize a data point within a time interval. This could be due to the data's granularity being directly proportional to its volume. Consequently, more parameters will be required in neural networks comprising highly granular data.

Volume: Although the volume of the data closely relates to the frequency of the data and the specific unit of data (Prado, 2018), we should differentiate volume from frequency because, while a high frequency typically translates to a relatively high volume, volume size might not directly correlate to the data frequency. This becomes more apparent when we consider seasonality or holidays for the same time interval. We can also recognize that, based on the time of day, the volume of data generated for the same subject of interest within the same period could be vastly different, suggesting a differential occurrence rate. This is particularly relevant for non-conventional data types, such as news and social media data, where high volume (i.e., the size of the volume) might not be directly correlated to data frequency. This becomes more apparent when we consider seasonality or holidays for the same time interval. We can also notice that based on the time of day, the volume of data generated for the same subject of interest within the same period could be vastly different, suggesting a different rate of occurrence. This is particularly relevant for non-conventional data types, such as news or social media data.

Using Apple Inc. as an example (Investing.com, 2013), a day marking a product announcement produces a substantially larger volume of news articles and relevant social media content than other days. Although this content might not affect the volume of the trading data—which depends more heavily on market data frequency—such instances might produce noticeable differences in the rate of change in market values. An increased rate warrants a different level of attention compared to a typical market day. The relationship between market data frequency and alternative data volume itself represents an interesting area of research that deserves a special level of attention.

Understanding data volume and data frequency is critical to designing infrastructure for processing data. As data volume approaches the realm of big data, precluding efficient computation in memory, it is necessary to consider alternative ways of processing data while utilizing relevant components of that data. Here, we begin considering ways of parallelizing the learning process without losing relationships between parallel batches. Data processing at such a scale requires parallel processing tools, such as those described by (Zaharia et al., 2010).

2.2.2 Data types

Market Data: Market data are trading activities data generated by trading venues such as stock exchanges and investment firms. They are typically provided via streaming data feeds or application programming interface (API) used within protocols such as the financial information exchange (FIX) (Prado, 2018). A typical trade message concerning stock market data comprises a ticker symbol (representing a particular company), bid price, ask price, time of last quote, and size of the sale.

Ticker symbol	AAPL
Name	Apple Inc.
Last trade price	289.80
Last trade timestamp	1577480401
Last trade volume	35447203
Exchange	NASDAQ

Table 2.1: A sample trade message for Apple Inc. (AAPL)

For messages with quote data, we should expect to see both the *bid* price & volume and the *ask* price & volume. These represent how much people are willing to buy and sell the asset at a given volume. Market data represent the core data type used by machine learning research in the stock market context and typically provide a detailed representation of trading activities regarding market assets such as equities/shares, currencies, derivatives, commodities, and digital assets. Derivatives can be further broken down into futures, forwards, options, and swaps (Bacon, 2023).

Market data can be either real-time or historical (Prado, 2018). Real-time data are used to make real-time trading decisions about buying and selling market instruments. Historical data are used to analyze historical trends and make informed decisions regarding future investments. Typically, historical data can contain intraday or end-of-day data summaries. The granularity of real-time data can be as detailed as a fraction of a second, with some tolerance for short delays. Comparing data for the same period, the frequency of a real-time data feed is expected to be much higher than historical data.

We can further separate market data, based on the details it contains, into Level I and Level II market data. Level II data contains more information and provides detailed information on bids and offers at prices other than the highest price (Z. Zhang et al., 2019). Level I data generally contain the basic trading data discussed thus far. Level II data are also referred to as order book or depth of book because they show details of orders that have been placed but not yet filled. These data also show the number of contracts available at different bid and ask prices.

Date	Time	Open	High	Low	Close	Volume
20160128	10:00	122.17	122.27	122.09	122.09	4,934
20160128	11:00	121.42	121.60	121.38	121.52	12,254

Table 2.2: Intraday time bar for ticker IBM

Fundamental data: Unlike market data, where data directly relate to trading activity on the asset of interest, fundamental data are based on information about the company the asset is attached to (Prado, 2018). Such data depict the company's standing using information such as cash flow, assets, liabilities, profit history, and growth projections. This kind of information can be obtained from corporate documentation such as regulatory filings and quarterly reports. Care has to be taken to confirm whether fundamental data points are publicly available because these are typically reported with a lapse. This means that analyzing the data must align properly with the date it became publicly available and not necessarily the date the report was filed or indexed.

Notably, some fundamental data are reported with some data yet to be made available, becoming backfilled upon availability. When fundamental data are published before source data becomes available, placeholder values are used during the interim period. Furthermore, given companies can issue revisions or corrections to sources multiple times, these will need to be corrected in the fundamental data, which suggests the need to incorporate a backfilling technique into the data consumption design. By definition, the frequency of this kind of data is very low compared to market data. This might explain why limited deep learning literature employs fundamental data. However, this also indicates the existence of a gap in research utilizing this kind of data, which would ideally be filled by considering fundamental data alongside other data types to provide a significant learning signal that remains to be fully exploited.

Alternative Data: Alternative data represents any other unconventional data type that can add value to already-established sources and types (Prado, 2018). This can range from user-generated data (e.g., social media posts, financial news, and comments) to Internet-of-Things data (e.g., data from different sensors and devices). Alternative data typically complement the aforementioned data types, especially market data. Given the nature of alternative data, they are typically much larger, hence requiring a sophisticated processing technique.

Notably, alternative data includes a vast amount of data that is open to interpretation because the signal might not be immediately obvious. For example, a market participant interested in Apple Inc. stocks might choose to observe different news articles related to the

company. Although there might be no direct reports about the company releasing a new product line, news reports about key meetings or large component purchases can indicate the plausibility of action. Accordingly, stock market professionals and researchers have become attentive to such indirect signals, and now consider alternative data essential to their data pipeline. Numerous researchers now combine traditional data types with either or both news articles and social media content to make market predictions. Social media especially has become a very popular alternative data type, primarily due to its position in the mainstream.

Market data attributes	Fundamental data attributes	Alternative data attributes
open price, high price, low price, close price, volume	revenue, earnings per share, market capitalization, dividend, average volume, shares outstanding, next earning date	google trends, news, texts, tweets, satellite imagery

Table 2.3: Representative attributes by data types

Table 2.3 presents certain representative attributes of the different data types. All of the attributes associated with market data and fundamental data are numerical and aggregated based on the available time series. For example, the intraday market data entry in row 1 of Table 2.2 shows the open and close prices for a one-hour time window that begins at 10 am and ends at 10:59 am. It also includes the maximum and minimum price and the total volume traded within the same window.

A fourth data type known as *Analytics data* (Prado, 2018), describes data derived from any of the other three types. Attributes of analytics data are earnings projections or sentiments from news or tweets that are combined with trade volume. We have chosen not to include this category because it does not clearly represent a direct source, and it is usually unclear what heuristics have been used to obtain the derived data points. Furthermore, given the objective of academic research is to make the metrics explicit, it is counter-intuitive to consider them useable input.

Table 2.4 presents the characteristics of the data employed by the literature reviewed, including the aforementioned data types. It is apparent that market data represents the most common type, with actual trading prices and volumes often paired with fundamental data to compute technical indicators (Soleymani and Paquet, 2020; Jingyuan Wang et al., 2019). Table 2.5 presents a more complete representation of freely or publicly available data sources that fully itemizes attributes.

Sources including `investing.com`, `finance.yahoo.com` and `kaggle.com` utilize either

Source	Type	Frequency	Free	Library
investing.com	market, fundamental	interday	Y	investpy
wrds-www.wharton.upenn.edu*	market, fundamental	interday, intraday	N	na
bloomberg.com	market, fundamental	interday, intraday	N	na
finance.yahoo.com	market, fundamental	interday, intraday	Y	yfinance
kaggle.com	market	interday**	Y	kaggle-api
interactivebrokers.com	market, fundamental	interday, intraday	N	tws-api
www.taifex.com.tw	taiwan market	interday, intraday	Y	na
pypi.org/project/tushare	china market, fundamental	interday, intraday	Y	tushare
optionmetrics.com	market, fundamental	interday, intraday	N	na
refinitiv.com	market, fundamental	interday, intraday	N	na
datashop.deutsche-boerse.com	market, fundamental	interday, intraday	N	na
trkd.thomsonreuters.com	market, fundamental	interday, intraday	N	na
www.wind.com.cn	china market, fundamental	interday, intraday	N	na
etsin.fairdata.fi	nordic market	intraday**	Y	na
londonstockexchange.com	UK market	interday	N	na
pinnacledata2.com	market, fundamental	interday, intraday	N	na
www.apex.com.tw	taiwan market, fundamental	interday, intraday	N	na
joinquant.com	china market, fundamental	interday, intraday	N	jqdatasdk

∗ compustat daily updates ∗∗ subject to availability

Table 2.4: Characteristics of data in survey

API or libraries, facilitating interactions with them and unlocking better integration with the machine learning system. Sources without any programmatic interface usually make data available as manual downloadable files.

The other major factor that affects the preferred data source is the frequency of availability, for example, whether the data is available multiple times a day (intraday data) or once a day (interday data). Given the potential volume and size of historical data, it is common for intraday data to remain available for a shorter timeframe than interday data, especially for freely available data sets. However, in most cases, it is possible to pay for intraday data for a longer timeframe if required for lower latency projects.

2.2.3 Data representation

Data generated from the stock market are typically represented as *Bars* and *Charts*. It is worth discussing these representations because they represent the most typical forms of representing data either numerically (bars) or graphically (charts).

Source	Market data attributes	Fundamental data attributes	Frequency
investing[1]	open price, high price, low price, close price, volume	revenue, earnings per share, market capitalization, dividend, average volume, ratio, beta, shares outstanding, next earning date	daily, weekly, monthly
y-finance[2]	open price, high price, low price, close price, volume	major holders, institutional holders, mutual fund holders, dividends, splits, actions, calendar, earnings, quarterly earnings, financials, quarterly financials, balance sheet, quarterly balance sheet, cashflow, quarterly cashflow, sustainability, shares outstanding	1 min, 2 mins, 5 mins, 15 mins, 30 mins, 60 mins, 90 mins, 1 hour, 1 day, 5 days, 1 week, 1 month, 3 months
taifex[3]	open bid, high bid, low bid, last bid, volume, best bid, best ask, historical high, historical low	not available	daily
kaggle[4]	open price, high price, low price, close price, volume	not available	daily
tushare[5]	open price, high price, low price, close price, volume	account receivable turn day, account receivable turnover, business income, current asset days, current asset turnover, earnings per share, earnings per share (year over year), fixed assets, gross profit rate, inventory days, inventory turnover, liquid assets, net profit ratio, net profits, outstanding, profits (year over year), report date, reserved, reserved per share, return on equity, time to market, total assets	daily
etsin[6]	open price, high price, low price, close price, volume	not available	daily

Table 2.5: Characteristics of public data sources

Bars

Bars enable the extraction of valuable information from market data in a regularized manner (Prado, 2018). They categorize futures into standard and more advanced types, with the advanced types comprising derivative computation from standard types. However, standard types are more common and also form the basis of chart representation.

Standard bars help to summarize market data into equivalent intervals and can be used with both intraday and historical data. The different types of standard bars all typi-

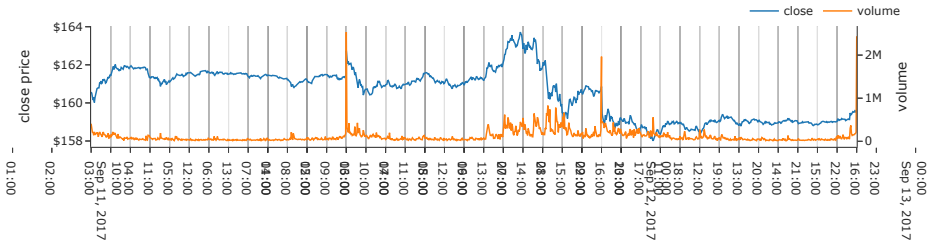


Figure 2.1: Intraday tick time series showing trade price and volume within the trading hours, across 2 days (Investing.com, 2013)

cally contain certain basic information for the specified interval, including the *timestamp, volume-weighted average price (VWAP), open price, close price, high price, low price*, and *traded volume*, all within the specified interval. The VWAP is based on the total traded for the day, irrespective of the time interval, and is computed as $\sum price \cdot volume / \sum volume$. The different standard bars are described in the following paragraphs.

Time bars: This is the most common bar type and derives from summarizing data into an equivalent time interval that includes all of the aforementioned standard bar information. Intraday hourly time bars feature hourly standard bar information for every hour of the day. For historical data, it is common to obtain details for each day. Table 2.2 exemplifies intraday time bars that can capture information.

The VWAP assists by demonstrating the trend for the price of a traded item during a given day. This single-day indicator is reset at the start of each trading day and should not be used in the context of daily historical data.

Tick bars: Unlike time bars that capture information at regular time intervals, tick bars capture the same information at a regular number of transactions or *ticks*. Ticks are trades in the stock market that can be used to represent the movement of price in trading data (i.e., the *uptick* and *downtick*). Ticks are commonly used for different stages of modeling market data, as in the case of *backtesting*. However, historical stock market data are not as freely accessible in the form of tick bars, especially for academic research purposes. For this purpose, most of the literature reviewed uses time bars, despite its statistical inferiority for predictive purposes.

Volume bars: Although tick bars exhibit better statistical properties than time bars (i.e., they are closer to independent distribution), they still feature the shortcoming of

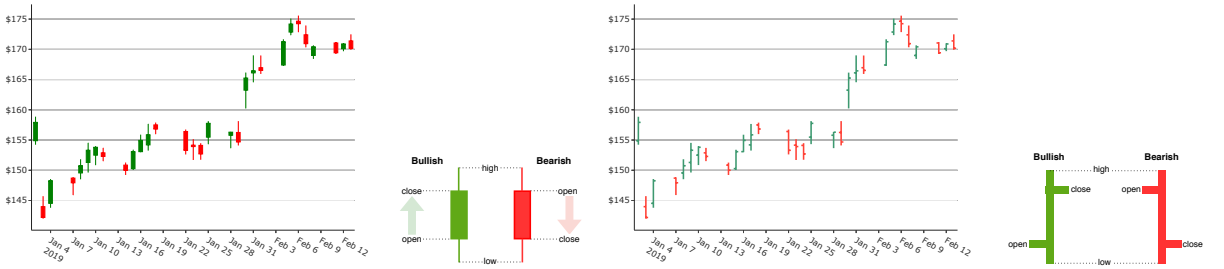
uneven distribution and propensity for outliers (Prado, 2018). This can be because a large volume of trade is placed together from accumulated bids in the order book, which gets reported as a single tick, or because orders are equally recorded as a unit, irrespective of size. That is, an order for 10 shares of a security and an order for 10,000 shares are both recorded as a single tick. Volume bars help to mitigate this issue by capturing information at every predefined volume of securities. Although volume bars feature better statistical properties than tick bars (Easley et al., 2012), they are similarly seldom used in academic research.

Range bars: Range bars involve information being captured when a predefined monetary range is traded. They are also referred to as dollar bars (Prado, 2018). Range bars are particularly useful because, by nature, securities appreciate or depreciate constantly over a given period. Consider a security that has depreciated by 50% over a certain period; by the end of that period, it is possible to purchase twice as much as at the beginning. For instance, consider a security that has depreciated from \$100 to \$50 over a given period. A capital investment of \$1000 would only have obtained 10 units at the start of the depreciation period; however, at the end of the period, that investment can obtain 20 units. Furthermore, corporate actions (e.g., splits, reverse splits, and buy-backs) do not impact range bars to the extent that they impact ticks and volume bars.

Charts

Charts visually represent the aforementioned bars, especially time bars. It might not be clear how these are relevant to a survey of deep learning applications in the stock market context, given it is possible to use the actual data that the charts are based on. However, various novel applications have used charts as training data. For example, (Kusuma et al., 2019) uses the candlestick plot chart as the input image for a convolutional neural network (CNN) algorithm. The charts most commonly used to visually represent stock market data are line, area, bar, and candlestick charts. Of interest here, however, are the candlestick and bar charts, which visually encode valuable information that can be used as input for deep learning algorithms.

Candlestick and bar charts can visually represent *open-high-low-close (OHLC)* data, as Figure 2.2 shows. These two types of charts are optionally color-coded, with red indicating bearish (closing lower than it opened) and green indicating bullish (closing higher than it opened). By properly encoding this information into these charts, an algorithm such as CNN can interpret numerous signals to generate an intelligent model.



(a) candlestick chart (b) candlestick description (c) bar chart (d) bar description

Figure 2.2: Candlestick & Bar Charts

2.3 Deep learning in finance

2.3.1 What is deep learning?

Recall that deep learning describes a machine learning technique based on networks of simple concepts and featuring different arrangements or architecture that allows computers to learn complicated concepts from simple nodes that are graphically connected using multiple layers (Goodfellow et al., 2016). The resurgence of deep learning was led by probabilistic or Bayesian models such as deep belief networks (DBN) (Goodfellow et al., 2016; Z. Hu et al., 2021), which comprise nodes representing random variables with probabilistic relationships to each other. More recently, however, artificial neural networks (ANN) that comprise nodes representing neurons that are generated by the training process have witnessed increasing popularity. All of the architectures we encounter in this survey are based on ANN; this section details these architectures.

Generally speaking, ANN are information processing systems with designs based on the human nervous system, specifically the brain, and that emphasize problem-solving (Castro, 2006). Typically, they comprise many simple processing elements with adaptive capabilities that can process a massive amount of information in tandem. Given neurons are the basic units for information processing in the brain, their simplified abstraction forms the foundation of ANN. The features and performance characteristics that ANN share with the human nervous system are (Castro, 2006):

1. The initial information processing unit occurs in elements known as *neurons*, *nodes* or *units*.

2. Neurons can send and receive information from both each other and the environment.

3. Neurons can be connected, forming a connection of neurons that can be described as *neural networks*.

4. Information is transmitted between neurons via connection links called *synapses*.

5. The efficiency of synapses, represented by an associated *weight* value or *strength*, corresponds, in aggregate, to the information stored in the neural network.

6. To acquire knowledge, connective strengths (aggregated weight values) are adapted to the environmental stimuli, a process known as *learning*.

Patterns are created by the information stored between neurons, which represents their synaptic or connective strength (Goodfellow et al., 2016). Knowledge is represented to influence the course of processing, which becomes a part of the process itself. This invariably means that learning becomes a matter of finding the appropriate connective strength to produce satisfactory activation patterns. This generates the possibility that an information processing mechanism can learn by tuning its connective strength during the processing course. This representation also reveals that knowledge is distributed over the connections between numerous nodes, meaning no single unit is reserved for any particular pattern.

Thus, an ANN can be summarized according to these three key features:

1. A set of *artificial neurons*, also known as nodes, units, or neurons.

2. A method for determining weight values, known as *training* or *learning* techniques.

3. A pattern of connectivity, known as the *network architecture* or *structure*.

The following sections detail these three features.

Artificial neurons

A biological neuron primarily comprises a *nucleus* (or *soma*) in a *cell body* and *neurites* (*axons* and *dendrites*) (Castro, 2006). The axons send output signals to other neurons, and the dendrites receive input signals from other neurons. The sending and receiving of signals take place at the *synapses*, where the sending (or *presynaptic*) neuron contacts the receiving (or *postsynaptic*) neuron. The synaptic junction can be at either the cell body

or the dendrites. This means that the synapses are responsible for signal/information processing in the neuron, a feature that allows them to alter the state of a postsynaptic neuron, triggering an electric pulse (known as *action potential*) in that neuron. The spikes cause the release of neurotransmitters at the axon terminals, which form synapses with the dendrites of other neurons. The action potential only occurs when the neuron's intrinsic electric potential (known as *membrane potential*) surpasses a threshold value.

An *artificial neuron* attempts to emulate these biological processes. In an artificial neuron, the synapse that connects the input to the rest of the neuron is known as a *weight*, characterized by *synaptic strength, synaptic efficiency, connection strength*, or *weight value*. Figure 2.3 show a typical artificial neuron.

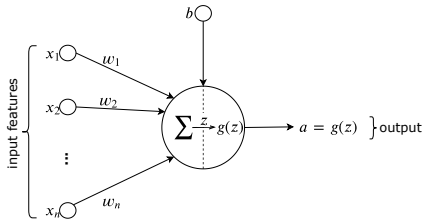


Figure 2.3: Model of a typical neuron (Castro, 2006)

As each input connects to the neuron, it is individually multiplied by the synaptic weight at each of the connections, which are aggregated in the *summing junction*. The summing junction adds the product of all of the weighted inputs with the neuron's bias value, i.e., $z = \sum \mathbf{w}\mathbf{x} + b$. The images essentially represent this. The *activation function* (also referred to as the *squashing function*) is represented as $g(z)$ and has the primary role of limiting the permissible value of the summation to some finite value. It determines a neuron's output relative to its net input, representing the summing junction's output. Thus, the neuron's consequent output, also known as the *activation* (a), becomes:

$$a = g(z) = g\left(\sum_{j=1}^{n} w_j x_j + b\right) \tag{2.1}$$

During the learning process, it is common to randomly initialize the weights and biases. These parameters are used by the activation to compute the neuron's output. In this simple representation of one neuron, we can imagine that the output (prediction) of the neuron is compared with the input (true value) using a *loss function* to generate the error rate. Through an optimization method called *Stochastic Gradient Descent*, the error rate is propagated back to the network, a process called *backpropagation* (Rumelhart et al.,

1986). This process is repeated over multiple iterations or *epochs* until a defined number of iterations is achieved or the error rate falls below a satisfactory threshold.

Multiple types of activation functions (Russell and Norvig, 2021) are used across different neural network architectures. The rectified linear unit (ReLU) activation function has been more popular in recent applications of feed-forward neural network (FFNN) because it is not susceptible to the vanishing gradient issue (Russell and Norvig, 2021), which impacts the use of the sigmoid function across multiple layers. It is also more computationally efficient. Other ReLU generalizations, such as Leaky ReLU or parametric ReLU (PReLU) are also commonly used. However, sigmoid continues to be used as a gating function in recurrent networks to maintain values between 0 and 1, hence controlling what passes through a node (Goodfellow et al., 2016). The hyperbolic tangent (tanh) activation function is also commonly used in recurrent networks, keeping the values that pass through a node between -1 and 1 (Goodfellow et al., 2016).

Learning techniques

In the ANN context, learning refers to the way a network's parameters adapt according to the input data. Typically, the learning technique is based on how weights are adjusted in the network and how data is made available to the network.

- Technique based on weight adjustment: The most common learning technique category, this technique is based solely on how weights are adjusted across an iterative process and is dependent on the type of supervision available to the network during the training process. The different types are supervised, unsupervised (or self-organized), and reinforcement learning.

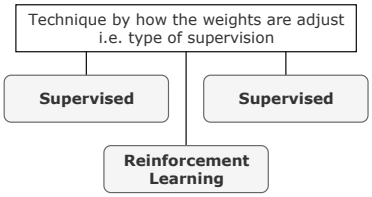


Figure 2.4: Supervision-based learning technique

- Technique based on data availability: When categorized according to how data is presented to the network, the learning technique can be considered offline or online. This technique might be chosen because the complete data are not available for training in one

batch. This could be because either data are streaming or a concept in the data changes at intervals, requiring the data to be processed in specific time windows. Another reason could be that the data are too large to fit into the memory, demanding processing in multiple smaller batches.

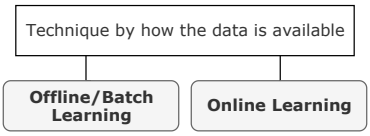


Figure 2.5: Learning technique based on data availability

Techniques based on supervision are most common for deep learning (and indeed deep learning), with increasing studies adopting batch learning approaches. Nonetheless, the primary architecture of deep learning networks is not exclusive to one technique category; instead, it is typical to find a mix of both, i.e., offline supervised learning and online reinforcement learning. Unless otherwise specified, it can be assumed that the technique is offline/batch learning. For example, *supervised learning* refers to *offline supervised learning* unless it is specified as online. The key point is that each supervision-based technique can be further categorized according to data availability.

Network architecture

The architecture of an ANN importantly contributes to the ways that it is organized. Network inputs depend solely on training data, and, for the most part, the output represents a function of the expected output. The layers between the input and output are mostly a design decision that depends largely on the network architecture, which is based on a typical neural network's system of multiple connections. Numerous ANN architectures exist across various domains, including communication systems and healthcare (Aceto et al., 2019; O'Shea and Hoydis, 2017; Xiao, 2021), with the stock market applications this survey considers adopting even more derivative architectures with easily identifiable and well-known foundations. Figure 2.6 presents these architectures and their common categorizations based on how they learn weight parameters. The following section describes their differences.

The learning techniques based on these architectures can be either *discriminative* or *generative*. A discriminative model discriminates between different data classes by learning the boundaries between them or the conditional probability distribution $p(y|x)$; meanwhile,

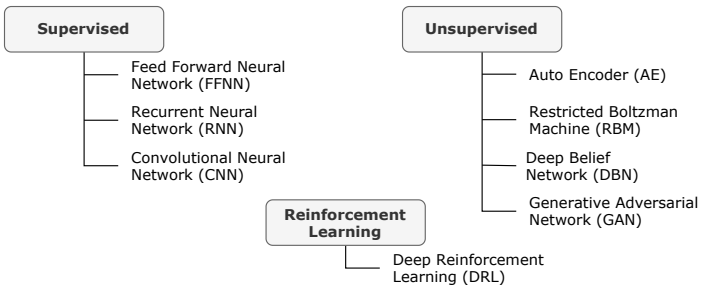


Figure 2.6: Taxonomy of deep learning architecture used in stock market applications

a generative model learns the distribution of individual classes or joint probability distribution $p(x, y)$ (Hinton, 2017). Although most traditional ANN architectures are discriminative, *autoencoders* and *BoltzMann machine* are considered generative. In a *Generative Adversarial Network* (Hinton, 2017), the two techniques are combined in a novel adversarial manner.

Feed-Forward Neural Networks: Comprising multiple neurons connected in layers, deep learning architectures use FFNN widely. Figure 2.7 presents the architecture of an FFNN. It comprises an *input layer*, representing the input example, one or more *hidden layers*, and an *output layer* (Goodfellow et al., 2016).

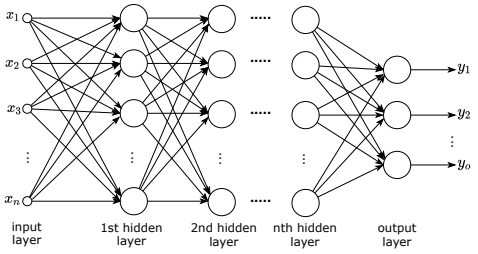


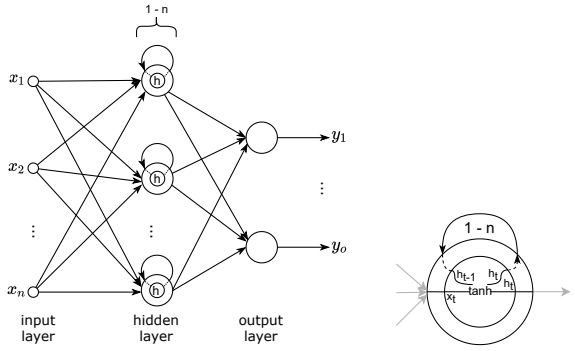
Figure 2.7: n-layer feed-forward neural network (Castro, 2006)

Although Goodfellow et al. (2016) suggest that *"a single layer is sufficient to represent a function",* hey also recommend deeper layers for better generalization. Ideally, the number of hidden layers should be decided for the specific task via experimentation. The input layer comprises a feature vector representing the input example that is fed to the first hidden layer. The hidden layer(s) and the output layer comprise multiple neurons, each with a vector of weights of the same size as the input, as well as a bias value. Within the

layers, each neuron's output becomes the input for the next layer, until, finally, the output layer uses the final activation to represent the model's prediction.

Broadly, this process aims to derive a generalization about the weights and biases associated with each neuron in the network, that is, derive generalizable values of $\mathbf{w}, b$ to compute $z = \sum \mathbf{wx} + b$ for each neuron (with input $\mathbf{x}$) in the network. Using an iterative training process of forward and backward propagation over multiple examples (training data), each layer's activations are propagated forward across the network, and the error rate is propagated back to the first hidden layer. Following the learning process, the network (model) can then be used to predict unseen/untested examples.

Recurrent Neural Network: recurrent neural network (RNN) are a special type of neural network that keeps a representation of the previously seen input data. These networks are ideal for processes where the temporal or sequential order of the input example is relevant (Goodfellow et al., 2016).



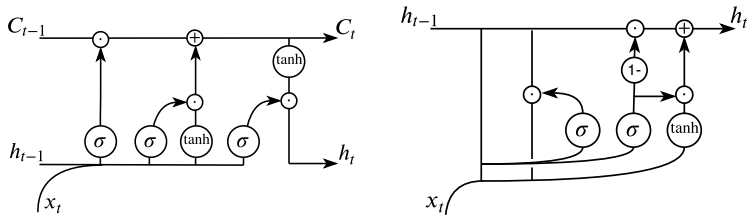
(a) RNN (b) Closer look at a RNN cell

Figure 2.8: RNN (Goodfellow et al., 2016)

The recurrence is represented as a loop in each neuron, as Figure 2.8 shows, allowing one or more passes of the same input, with the network maintaining a state representation of each pass. Following the specified number of passes, the final state is transmitted as output parameters. This means that RNN allows the possibility of inputs and outputs of variable length. That is, given the loop's flexibility, the architecture can be constructed to be one-to-one, one-to-many, many-to-one, or many-to-many.

However, typical RNN, make it difficult for the hidden state to retain information over a long period. That is, they have a short memory due to the gradient becoming

smaller and smaller as it is propagated backward in time steps across the recurring loop, a phenomenon known as *vanishing gradient*. This means that for temporal data, in which the relevant relationship between data points occurs over a lengthy period, a typical RNN model is not ideal. Thus, other versions of RNN have been formulated, with the most frequently used approaches being long short-term memory (LSTM) and gated recurrent unit (GRU) (Goodfellow et al., 2016). The architectures discussed can largely reduce the vanishing gradient effect by maintaining a cell state via additive updates rather than just the RNN hidden state with product updates.



(a) LSTM (b) GRU

Figure 2.9: LSTM & GRU (Goodfellow et al., 2016)

Convolutional Neural Networks: Another network architecture type that has gained substantial popularity, especially for analyzing digital images, is CNN (Goodfellow et al., 2016). The reason is that CNN can simplify large amounts of pixel density, vastly reducing the number of parameters to work with, making the ANN highly efficient. Unlike more conventional ANN, in which the input is represented as a feature vector, CNN represent the input as a matrix, which they use to generate the first *convolutional layer*.

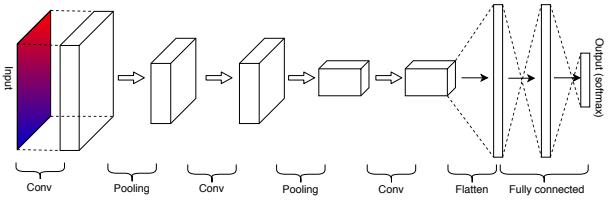


Figure 2.10: Architecture of a convolutional neural network (Goodfellow et al., 2016)

A typical CNN will contain one or more convolutional layers, each connected to its respective *pooling layer*. Figure 2.10 provides a simple representation of such a network.

Autoencoder: Autoencoders are unsupervised ANN that efficiently encode input data, a process known as *latent representation* or *encoding*. This process involves using input data as a feature vector and attempting to reconstruct the same data using fewer nodes than the input (Goodfellow et al., 2016). As such, autoencoders are frequently used for dimensionality reduction.

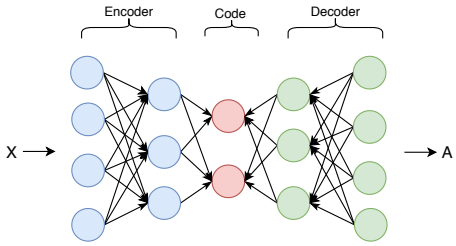


Figure 2.11: A simple Autoencoder (Goodfellow et al., 2016)

As Figure 2.11, shows, an autoencoder's architecture imposes a bottleneck for encoding the input representation. A decoder layer subsequently reproduces an output to represent the reconstructed input. In so doing, it learns a representation of the input data while ignoring the input noise. The encoder's representation of the transformed input is referred to as the emphcode, code, and it is the internal or hidden layer of the autoencoder. The decoder subsequently generates the output from the code.

Autoencoders are commonly used in stock market data for their dimension reduction functionality (L. Chen et al., 2018; Chong et al., 2017) to avoid dimensionality curse (Soleymani and Paquet, 2020). This is an important consideration for stock market data, where there is value in network simplicity without losing important features. In (Soleymani and Paquet, 2020), a restricted stacked autoencoder network reduces an 11-feature set to a 3-feature set before it is fed into a CNN architecture in a deep reinforcement learning framework called *DeepBreath*. This enables an efficient approach to a portfolio management problem in a setting that combines offline and online learning. Elsewhere, (G. Hu et al., 2018) combines CNN and autoencoder architectures in its *convoluted autoencoder (CAE)* to reduce candlestick charts to numerical representations to improve stock similarity.

Deep Reinforcement Learning: Unlike supervised and unsupervised learning, in which all learning occurs within the training dataset, a *reinforcement learning (RL)* problem is formulated as a discrete-time stochastic process. The learning process interacts with the *environment* via an iterative sequence of actions, state transitions, and rewards, in a bid to

maximize the cumulative reward (François-Lavet et al., 2018). The future state depends only on the current state and action, meaning it learns using a trial-and-error *reinforcement* process in which an *agent* incrementally obtains experience from its environment, thereby updating its current state. The action to take (from the action space) by the agent is defined by a *policy*.

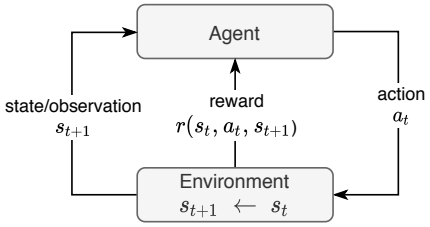


Figure 2.12: Reinforcement Learning (François-Lavet et al., 2018)

It is common to see a RL system formulated as a Markov decision process Markov decision process (MDP) in which the system is fully observable, i.e., the state of the environment is the same as the observation that the agent perceives (François-Lavet et al., 2018). Furthermore, RL can be categorized as *model-based* or *model-free* (Russell and Norvig, 2021).

- *Model-Based Reinforcement Learning*: The agent retains a transition model of the environment to enable it to select actions that maximize the cumulative utility. The agent learns a *utility function* that is based on the total rewards from a starting state. It can either start with a known model (i.e., chess) or learn by observing the effects of its actions.
- *Model-Free Reinforcement Learning*: The agent does not retain a model of the environment, instead focusing on directly learning how to act in different states. This could be via either an *action-utility function* (Q-learning) that learns the utility of taking an action in a given state or a *policy-search* in which a *reflex agent* directly learns to map policy, $\pi(s)$, from different states to corresponding actions.

deep reinforcement learning (DRL) is a deep representation of RL that can be model-based, model-free, or a combination of the two (Ivanov and D'yakonov, 2019). The stock market can be considered to feature an DRL characteristic, with past states well-encapsulated in current states and events and the only requirement for future states being

the current state. For this reason, DRL is a particularly popular approach for modern quantitative analysis of the stock market. Applications of DRL in these scenarios vary from profitable/value stock selection or portfolio allocation strategy (X. Li et al., 2019; Jingyuan Wang et al., 2019) to simulating market trades in a bid to develop optimal liquidation strategy (Bao and Liu, 2019).

2.4 Deep learning in the stock market

In Section 2.3.1, we considered what deep learning is and discussed certain specific deep learning architectures that are commonly used in stock market applications. Although we referred to certain specific uses of these network types that are employed in the stock market, it is important to note that all of the architectures mentioned are also commonly used for other applications. However, some specific considerations must be kept in mind when the stock market is the target. These range from the model's composition to backtesting and evaluation requirements and criteria. Some of these items do not correspond to a traditional machine learning toolbox but are crucial to stock market models and cannot be ignored, especially given the monetary risks involved.

This section first discusses the specifics of modeling considerations for stock market applications. It also discusses backtesting as an integral part of the process and details some backtesting methodology. This is followed by a review of the different evaluation criteria and evaluation types.

Modelling considerations

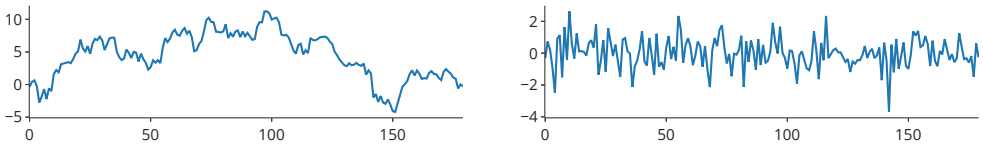
When training an machine learning model for most applications, we consider how *bias* and *variance* affect the model's performance, and we focus on establishing the tradeoffs between the two. Bias measures how much average model predictions differ from actual values, and variance measures the model's generalizability and its sensitivity to changes in the training data. High degrees of bias suggest underfit, and high levels of variance suggest overfit. It is typical to aim to balance bias and variance for an appropriate model fit that can be then applied to any unseen dataset, and most machine learning applications are tuned and focused accordingly.

However, in financial applications, we must exceed these to avoid some of the following pitfalls, which are specific to financial data.

Sampling intervals: Online machine learning applications typically feature sampling windows in consistent chronological order. While this is practical for most streaming data, it is not suitable for stock market data and can produce substantial irregularities in model performance. As Figure 2.1 demonstrates, the volume of trade in the opening and closing period is much higher than the rest of the day for most publicly available time-based market data. This could result from pre-market or after-hours trading and suggests that sampling at a consistent time will inadvertently undersample the market data during high-activity periods and undersample during low-activity periods, especially when modeling for intraday activities.

A possible solution is using data that has been provided in ticks, but these are not always readily available for stock market data without significant fees, potentially hindering academic study. Tick data can also make it possible to generate data in alternative bars, such as tick or volume bars, significantly enhancing the model performance. Notably, (Easley et al., 2012) uses the term *volume clock* to formulate volume bars to align data sampling to volume-wise market activities. This enables high-frequency trading to have an advantage over low-frequency trading.

Stationarity: Time-series data are either stationary or non-stationary. Stationary time-series data preserve the statistical properties of the data (i.e., mean, variance, covariance) over time, making them ideal for forecasting purposes (Prado, 2018). This implies that spikes are consistent in the time series, and the distribution of data across different windows or sets of data within the same series remains the same. However, because stock market data are non-stationary, statistical properties change over time and within the same time series. Also, trends and spikes in non-stationary time series are not consistent. By definition, such data are difficult to model because of their unpredictability. Before any work on such data, it is necessary to render them as stationary time series.



(a) Non-stationary time-series; $y_t = y_{t-1} + \epsilon_t$.

(b) Stationary time-series; $y_t = \epsilon_t$.

Figure 2.13: Time-series for the same value of $\epsilon_t \sim \mathcal{N}(0, 1)$

A common approach to converting non-stationary time series to stationary time series

involves differencing. This can involve either computing the difference between consecutive observations or, for seasonal time series, the difference between previous observations of the same season. This approach is known as *integral differencing*, with (Prado, 2018) discussing *fractional differencing* as a memory-preserving alternative that produces better results.

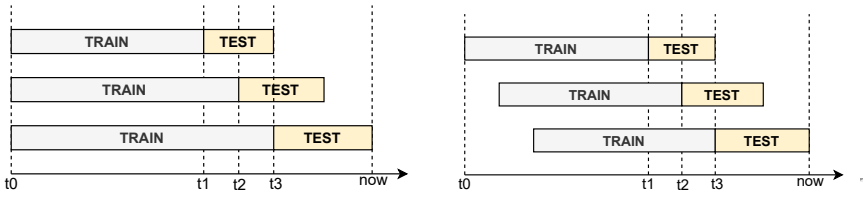
Backtesting: In machine learning, it is common to split data into training and testing sets during the modeling process. Given the goal of this exercise is to determine the accuracy or evaluate performance in some other way, it follows that adhering to such a conventional approach is appropriate. However, when modeling for the financial market, performance is measured by the model's profitability or volatility of the model. According to (Arnott et al., 2018), there should be a checklist or *Protocol* that mandates that machine learning research include the goal of presenting proof of positive outcomes through backtesting.

Opacity and bias in AI systems represent two of the overarching debates in AI ethics (Müller, 2020). Although a significant part of the conversation concerns the civil construct, it is clear that the same reasoning applies to other economic and financial AI applications. For example, (Müller, 2020) raises concerns about statistical bias and the lack of due process and auditing surrounding using machine learning for decision-making. This relates to conversations about honesty in backtesting reports and the selection bias that typically affects academic research in the financial domain (Fabozzi and De Prado, 2018).

In the context of deep learning in the stock market, backtesting involves building models that simulate trading strategy using historical data. This serves to consider the model's performance and, by implication, helps to discard unsuitable models or strategies, preventing selection bias. To properly backtest, we must test on unbiased and sufficiently representative data, preferably across different sample periods or over a sufficiently long period. This positions backtesting among the most essential tools for modeling financial data. However, it also means it is among the least understood in research (Prado, 2018).

When a backtested result is presented as part of a study, it demonstrates the consistency of the approach across various time instances. Recall that *overfitting* in machine learning describes a model performing well on training data but poorly on test or unseen data, indicating a large gap between the training error and the test error (Prado, 2018). Thus, when backtesting a model on historical data, one should consider the issue of *backtest overfitting*, especially during *walk-forward backtesting* (Prado, 2018).

Walk-forward is the more common backtesting approach and refers to simulating trading actions using historical market data—with all of the actions and reactions that might



(a) Expanding window (b) Sliding window

Figure 2.14: Backtesting strategies

have been part of that—in chronological time. Although this does not guarantee future performance on unseen data/events, it does allow us to evaluate the system according to how it would have performed in the past. Figure 2.14 shows two common ways of formulating data for backtesting purposes. Formulating the testing process in this manner removes the need for cross-validation because training and testing would have been evaluated across different sets. Notably, traditional K-fold cross-validation is not recommended in time series experiments such as this, especially when the data is not IID (Bergmeir and Benítez, 2012; Zaharia et al., 2010).

Backtesting must be conducted in good faith. For example, given backtest overfitting means that a model is overfitted to specific historical patterns, if favorable results are not observed, researchers might return to the model's foundations to improve generalizability. That is, researchers are not expected to fine-tune an algorithm in response to specific events that might affect its performance. For example, consider overfitting a model to perform favorably in the context of the 1998 recession, and then consider how such a model might perform in response to the 2020 COVID-19 market crash. By backtesting using various historical data or over a relatively long period, we modify our assumptions to avoid misinterpretations.

Assessing Feature Importance: In discussing backtesting, we have discussed why we shouldn't selectively "tune" a model to specific historical scenarios to achieve a favorable performance to challenge the usefulness of the knowledge gained from the model's performance in such experiments. *Feature Importance* becomes relevant here. Feature importance enables the measurement of the contribution of input features to a model's performance. Given neural networks are typically considered "black-box" algorithms, the movement around explanation AI contributes to the interpretation of the output of the network and understanding of the importance of the constituent features, as observed in the important role of *Feature Importance Ranking* in (Samek et al., 2017; Wojtas and K.

Chen, 2020). Unlike traditional machine learning algorithms, this is a difficult feat for ANN models, typically requiring a separate network for the feature ranking.

2.4.1 Model evaluation

Machine learning algorithms use evaluation metrics such as accuracy and precision. This is because we are trying to measure the algorithm's predictive ability. Although the same remains relevant for machine learning algorithms for financial market purposes, what is ultimately measured is the algorithm's performance with respect to returns or volatility. The works reviewed include various performance metrics that are commonly used to evaluate an algorithm's performance in the financial market context.

Recall that in Section 2.4 emphasized the importance of avoiding overfitting when backtesting. It is crucial to be consistent with backtesting different periods and to be able to demonstrate consistency across different financial evaluations of models and strategies. *Returns* represents the most common financial evaluation metric for obvious reasons. Namely, it measures the profitability of a model or strategy (Bacon, 2023). It is commonly measured in terms of rate during a specific window of time, such as day, month, or year. It is also common to see returns annualized over various years, which is known as *compound annual growth rate (CAGR)*. When evaluating different models across different time windows, higher returns indicate a better model performance.

However, it is also important to consider *Volatility* because returns alone do not relay the full story regarding a model's performance. Volatility measures the variance or how much the price of an asset can increase or decrease within a given timeframe (Bacon, 2023). Similar to returns, it is common to report on daily, monthly, or yearly volatility. However, contrary to returns, lower volatility indicates a better model performance. The volatility index (VIX), a real-time index from the Chicago Board Options Exchange (CBOE), is commonly used to estimate the volatility of the US financial market at any given point in time (Chow et al., 2021). The VIX measures the US stock market volatility based on its relative strength compared to the S&P 500 index, with measures between 0–12 considered low, measures between 13–19 considered normal, and measures above 20 considered high.

Building on the information derived from returns and volatility, the *Sharpe ratio* enables investors to identify little-to-no-risk investments by comparing investment returns with risk-free assets such as treasury bonds (Bacon, 2023). It measures average returns after accounting for risk-free assets per volatility unit. The higher the Sharpe ratio, the better the model's performance. However, the Sharpe ratio features the shortcoming of assuming the data's normal distribution due to the upward price movement. The*Sortino ratio* can

Evaluation	Description	Formula
Returns	Total amount gained or lost within a specific investment period, typically measured as a percentage of the original investment known as rate of returns (RoR) (Bacon, 2023). This could also be the absolute total profit or loss for the investment period.	$\frac{V_f - V_i}{V_i} * 100.$
compound annual growth rate (CAGR)	The RoR for investment over a number of years, with returns re-invested yearly (Bacon, 2023). n: Number of years.	$\left(\frac{V_f}{V_i}\right)^{\frac{1}{n}} - 1$
Volatility	Degree of variation in asset or total portfolio value (Bacon, 2023). σ: Standard deviation of returns; T: *Time Horizon* or number of holding period.	$\sigma\sqrt{T}$
Sharpe Ratio	Measures performance in comparison with a risk-free asset, with adjustments for volatility or total risk (Bacon, 2023). R_p: Average portfolio returns; r_f: Risk-free (i.e., treasury bonds) returns; σ_p: standard deviation of a portfolio's excess returns.	$\frac{R_p - r_f}{\sigma_p}$
Sortino Ratio	A modification of the Sharpe Ratio that differentiates harmful volatility from overall volatility (Bacon, 2023). σ_d: standard deviation of portfolio's negative returns, i.e., returns that fall below a user-defined threshold.	$\frac{R_p - r_f}{\sigma_d}$
maximum drawdown (MDD)	Measures the decline of a return from a peak before a new peak that is at least equal to the old peak is achieved (Bacon, 2023). This is used to compare the riskiness of different models or strategies. V_t: Trough value; V_p: Peak value.	$\frac{V_t - V_p}{V_p}$
Calmar Ratio	Risk-adjusted returns (Bacon, 2023).	$\frac{V_f - V_i}{MDD}$
value at risk (VaR) Threshold	Estimate (as threshold) of maximum loss for an investment over time (Bacon, 2023). E_r: Expected returns; z_i: z-score of confidence interval; σ_p: Standard deviation of portfolio; V_p: Value of portfolio.	$[E_r - (z_i * \sigma)] * V_p$

Table 2.6: Financial evaluation metrics

mitigate against this, differing by using only the standard deviation of the downward price movement rather than the full swing that the Sharpe ratio employs.

Other commonly used financial metrics are *MDD* and the *Calmar ratio*, both of which are used to assess the risk involved in an investment strategy. Maximum drawdown describes the difference between the highest and lowest values between the start of a decline in peak value to the achievement of a new peak value, which indicates losses from past

investments (Bacon, 2023). The lower the MDD, the better the strategy, with zero value suggesting zero loss in investment capital. The Calmar ratio measures the MDD adjusted returns on capital to gauge the performance of an investment strategy. The higher the Calmar ratio, the better the strategy.

Another metric considered important by the works reviewed was VaR, which measures risk exposure by estimating the maximum loss of an investment over time using historical performance (Bacon, 2023).

Meanwhile, other well-known non-financial machine learning metrics commonly used are based on the accuracy of a model's prediction. These metrics are calculated in terms of either the following *confusion matrix* or in terms of the difference between the derived and observed target values.

		Predicted		
		Positive	Negative	Total
Actual	Positive	TP	FP	P
	Negative	FN	TN	N
	Total	P'	N'	$P + N$

Table 2.7: Confusion matrix

True Positive (TP) and True Negative (TN) are the correctly predicted positive and negative classes respectively. Subsequently, False Positive (FP) and False Negative (FN) are the incorrectly predicted positive and negative classes (Han et al., 2012).

The evaluation metrics in Table 2.8 are expected to be used as complementary metrics to the primary and more specific financial metrics in Table 2.6. This is because the financial metrics can evaluate various investment strategies in the context of backtested data, which the machine learning metrics are not designed for. Section 2.4.3 shows how these different evaluation metrics are combined across the works of literature that we reviewed.

2.4.2 Applications of deep learning in the stock market

In this section, we set out to investigate applications of deep learning in the stock market context by answering three overarching research questions:

Question 1 What current research methods based on deep learning are used in the stock market context?

Evaluation	Description	Formula
Accuracy	The percentage of the correctly predicted classes.	$\frac{TP+TN}{P+N}$
Error rate	The percentage of incorrectly predicted classes. Also computed as $1 - accuracy$.	$\frac{FP+FN}{P+N}$
Recall (true predictive rate, tpr)	Ratio of true positive classes; also known as a measure of exactness or sensitivity.	$\frac{TP}{P}$
Precision (positive predictive value, ppv)	Ratio of positive predictions; also known as a measure of completeness.	$\frac{TP}{TP+FP}$
F-score	Harmonic mean of recall and precision.	$\frac{2 * ppv * tpr}{ppv + tpr}$
Weighted F-score	Weighted measure of recall and precision. $\beta < 1$ assigns more weight to precision, while $\beta > 1$ assigns more wait to recall.	$\frac{(1+\beta^2) * ppv * tpr}{\beta^2 * ppv + tpr}$; $\beta > 0$
mean absolute error (MAE)	Average of the absolute difference between the predicted values and the actual values.	$\frac{1}{n}\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$
mean absolute percentage error (MAPE)	Average of the percentage errors.	$\frac{100}{n}\sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i}$
mean square error (MSE)	Average of the squared difference between the predicted values and the actual values.	$\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Table 2.8: Machine learning evaluation metrics

Question 2 Are the research methods consistent with real-world applications, i.e., have they been backtested?

Question 3 Is this research easily reproducible?

Although many research works have used stock market data with deep learning in some form, we quickly discovered that many are not easily applicable in practice due to how the research has been conducted. Although we retrieved over 10,000 works[7], by not being directly applicable, most of the experiments are not formulated to provide insight for financial purposes, with the most common formulation being as a traditional machine

[7] Searched for *"deep learning" AND "stock market"* on Google Scholar

learning problem that assumes that it is sufficient to break the data into training and test sets.

Recall that we categorized learning techniques by data availability in Section 2.3.1. When the complete data are available to train the algorithm, it is defined as *offline* or *batch* learning. When that is not the case, and it is necessary to process the data in smaller, sequential phases, as in streaming scenarios or due to changes in data characteristics, we categorize the learning technique as *online*. Although machine learning applications in the stock market context are better classified as online learning problems, surprisingly, very few research papers approach the problem accordingly, instead mostly approaching it as an offline learning problem, a flawed approach (Prado, 2018).

To apply this approach to financial machine learning research for the benefit of market practitioners, the provided insight must be consistent with established domain norms. One generally accepted approach to achieving this is backtesting the algorithm or strategy using historical data, preferably across different periods (Bergmeir and Benítez, 2012; Institute, 2020). Although Section 2.4 discussed backtesting, we should re-iterate that backtesting does not constitute a "silver bullet" or a method of evaluating results. However, it does assist evaluation of the performance of an algorithm across different periods. Financial time-series data are not IID, meaning the data distribution differs across different independent sets. This also means that there is no expectation that results across a particular period will produce similar performances in different periods, no matter the quality of the presented result. Meanwhile, the relevant performance evaluation criteria are those that are financially specific, as discussed in Section 2.4.1. To this end, we ensured that the papers reviewed provide some indication of consideration of backtesting. An ordinary reference sufficed, even if the backtested results are not presented.

We used Google Scholar (Google, 2020) as the search engine to find papers matching our research criteria. The ability to search across different publications and the sophistication of the query syntax (Ahrefs, 2020) was invaluable to this process. While we also conducted spot searches of different publications and websites to validate that nothing was missed by our chosen approach, the query results from Google Scholar proved sufficient, notably even identifying articles that were missing from the results of direct searches on publication websites. We used the following query to conduct our searches:

> "deep learning" AND "stock market" AND ("backtest" OR "back test" OR "back-test")

This query searches for publications including the phrases ``deep learning'', ``stock market'', and any one of ``backtest'', ``back test'' or ``back-test''. We observed

these three different spellings of "backtest" in different publications, suggesting the importance of catching all of these alternatives. This produced 185 results[8], which include several irrelevant papers. For validation, we searched using Semantic Scholar (Scholar, 2020), obtaining approximately the same number of journal and conference publications. We chose to proceed with Google Scholar because Semantic Scholar does not feature such algebraic query syntax, requiring that we search for the different combinations of "backtest" individually with the rest of the search query.

Publisher	Count
IEEE	9
arXiv	8
SSRN	5
Elsevier	3
ACM	2
MDPI	2
Springer	2
IOP Publishing	1
Wiley	1
IJCAI	1
Institutional Investor Journals	1

Year	Count
2018	6
2019	10
2020	19

Table 2.9: Quantifying papers by publication and year of publication

The search query construct provided us with the starting point for answering research questions (1) and (2). Then, we evaluated the relevance to the research objective of the 185 publications and considered how each study answered question (3). We objectively reviewed all query responses without forming an opinion on the rest of their experimental procedure with the rationale that addressing the basic concerns of a typical financial analyst represents a good starting point. Consequently, we identified only 35 papers as relevant to the research objective. Table 2.9 quantifies the papers reviewed by publication and year of publication. It is interesting to observe the non-linear change in the number of publications over the last three years as researchers have become more conscious of some of these considerations.

2.4.3 Findings

Section 2.3.1 explained the different architectures of the deep ANN that are commonly used in stock market experiments. Based on the works reviewed, we can categorize the

[8] Query results as of November 5, 2020

algorithms into the following specializations:

- Trade Strategy: Algorithmically generated methods or procedures for making buying and selling decisions in the stock market.
- Price Prediction: Forecasting the future value of a stock or financial asset in the stock market. It is commonly used as a trading strategy.
- Portfolio Management: Selecting and managing a group of financial assets for long-term profit.
- Market Simulation: Generating market data under various simulation *what-if* market scenarios.
- Stock Selection: Selecting stocks in the stock market as part of a portfolio based on perceived or analyzed future returns. It is commonly used as a trading or portfolio management strategy.
- Risk Management: Evaluating the risks involved in trading, to maximize returns.
- Hedging Strategy: Mitigating the risk of investing in an asset by taking an opposite investment position in another asset.

Although a single specialization is usually the primary area of focus for a given paper, it is common to see at least one other specialization in some form. An example is testing a minor trade strategy in price prediction work or simulating market data for risk management. Table 2.10 illustrates the distribution of the different deep learning architectures across different areas of specialization for the studies reviewed by this survey. Architectures such as LSTM and DRL are more commonly used because of their inherent temporal and state awareness. In particular, LSTM is favorable due to its relevant characteristic of remembering states over a relatively long period, which price prediction and trade strategy applications, in particular, require. Novel use cases e.g., (Jingyuan Wang et al., 2019) combine LSTM and RL to perform remarkably well in terms of annualized returns. There are many such combinations in trade strategy and portfolio management, where state observability is of utmost importance.

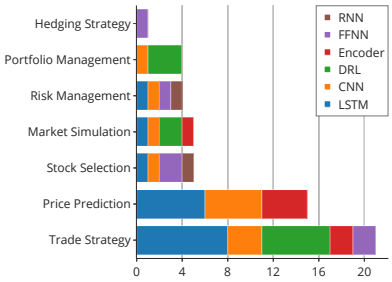
Although FFNN is seldom used by itself, there are multiple instances of it being used alongside other approaches, such as CNN and RNN. Speaking of CNN, it is surprising how popular it is, considering it is more commonly used for image data. True to its nature, attempts have been made to train models using stock market chart images (G. Hu et

Architecture	Count
LSTM	17
CNN	12
DRL	11
Autoencoder	7
FFNN	6
RNN	2

(a) Deep learning architecture. 1-n per publication

Specialization	Count
Trade Strategy	15
Price Prediction	8
Portfolio Management	4
Market Simulation	3
Stock Selection	3
Risk Management	1
Hedging Strategy	1

(b) Area of specialization. 1 per publication



(c) distribution of architectures across different specializations. 1-n architectures per specialization

Table 2.10: Quantifying the architectures and fields considered by publications surveyed

al., 2018; Kusuma et al., 2019). Given its ability to localize features, CNN is also used with high-frequency market data to identify local time series patterns and extract useful features (Chong et al., 2017). Autoencoders and restricted Boltzmann machine (RBM) are also used for feature extraction, with the output fed into another kind of deep neural network architecture.

A: architecture, B: market(s), C: dataset source, D: reproducibility

	Reference	A	B	C	D
	(Jingyuan Wang et al., 2019) Jingyuan Wang et al., 2019	DRL, LSTM	China, US	wind, wrds	No
Trade Strategy	(Y. Li et al., 2020) Y. Li et al., 2020	DRL	US	kaggle	No

A: architecture, B: market(s), C: dataset source, D: reproducibility

	Reference	A	B	C	D
	(Théate and Ernst, 2020) Théate and Ernst, 2020	DRL, FFNN	Asia, US, Europe	unspecified	Yes
	(Z. Zhang et al., 2020) Z. Zhang et al., 2020	DRL	US	pinnacle	No
	(Chakole and Kurhekar, 2020) Chakole and Kurhekar, 2020	DRL, FFNN	US, India	yahoo	No
	(J. Wu et al., 2019) J. Wu et al., 2019	DRL, LSTM	China	tushare	No
	(G. Hu et al., 2018) G. Hu et al., 2018	Autoencoder, CNN	UK	unspecified	No
	(Lei et al., 2020) Lei et al., 2020	CNN	China	tushare	No
	(Y. Y. Chen et al., 2018) Y. Y. Chen et al., 2018	CNN	Taiwan	apex	No
	(J. M. T. Wu et al., 2020) J. M. T. Wu et al., 2020	LSTM	Taiwan	tfe	No
	(Koshiyama et al., 2020) Koshiyama et al., 2020	Autoencoder, LSTM	Global	bloomberg	Yes
	(T. Sun et al., 2019) T. Sun et al., 2019	LSTM	US	ibkr	No
	(Silva et al., 2020) Silva et al., 2020	LSTM	Unspecified	unspecified	No
	(Jun Wang et al., 2020) Jun Wang et al., 2020	LSTM	China	joinquant	No
	(Chalvatzis and Hristu-Varsakelis, 2020) Chalvatzis and Hristu-Varsakelis, 2020	LSTM	US	unspecified	No
Price Prediction	(Jia Wang et al., 2019) Jia Wang et al., 2019	Conv-LSTM, RNN	China, US	ibkr	No
Price Prediction	(Z. Zhang et al., 2019) Z. Zhang et al., 2019	CNN, LSTM	UK, Nordic	lse, etsin	No
Price Prediction	(Zhao et al., 2018) Zhao et al., 2018	Autoencoder, CNN, LSTM	US	unspecified	No

A: architecture, B: market(s), C: dataset source, D: reproducibility

	Reference	A	B	C	D
	(H. Zhang, Q. Liang, Siheng Li, et al., 2020) H. Zhang, Q. Liang, Siheng Li, et al., 2020	Autoencoder, CNN, LSTM	China	unspecified	No
	(Fang et al., 2019) Fang et al., 2019	LSTM	China	private	No
	(Baek and Kim, 2018) Baek and Kim, 2018	LSTM	US	yahoo	Yes
	(J. Wang et al., 2018) J. Wang et al., 2018	CNN	US	unspecified	No
	(H. Zhang, Q. Liang, R. Wang, et al., 2020) H. Zhang, Q. Liang, R. Wang, et al., 2020	Autoencoder	China	unspecified	No
Portfolio Management	(Z. Liang et al., 2018) Z. Liang et al., 2018	DRL	China	investing, wind	Yes
	(Park et al., 2020) Park et al., 2020	DRL	Korea, US	investing, yahoo	No
	(Guo et al., 2018) Guo et al., 2018	DRL, CNN	China	unspecified	Yes
	(Jifei Wang and L. Wang, 2019) Jifei Wang and L. Wang, 2019	FFNN	US	bloomberg	No
Market Simulation	(Maeda et al., 2020) Maeda et al., 2020	DRL, LSTM, CNN	Simulated	none	No
	(Buehler et al., 2020) Buehler et al., 2020	Autoencoder	US	unspecified	Yes
	(Raman and Leidner, 2019) Raman and Leidner, 2019	DRL	US	trkd	No
Stock Selection	(C. Zhang et al., 2020) C. Zhang et al., 2020	FFNN	China	unspecified	No
	(Amel-Zadeh et al., 2020) Amel-Zadeh et al., 2020	RNN, FFNN	US	wrds	No
	(J. Yang et al., 2019) J. Yang et al., 2019	CNN, LSTM	China	unspecified	No

A: architecture, B: market(s), C: dataset source, D: reproducibility

	Reference	A	B	C	D
Risk Management	(Arimond et al., 2020) Arimond et al., 2020	CNN, FFNN, LSTM, RNN	EU, UK, US	refinitive	No
Hedging Strategy	(Ruf and W. Wang, 2020) Ruf and W. Wang, 2020	FFNN	EU, US	optionm, datashop	Yes

investing:investing.com **wrds:**wrds-www.wharton.upenn.edu **bloomberg:**bloomberg.com
yahoo:finance.yahoo.com **kaggle:**kaggle.com **ibkr:**interactivebrokers.com **tfe:**taifex.com.tw
tushare:pypi.org/project/tushare **optionm:**optionmetrics.com **refinitive:**refinitiv.com
datashop:datashop.deutsche-boerse.com **trkd:**trkd.thomsonreuters.com **wind:**wind.com.cn
etsin:etsin.fairdata.fi **lse:**londonstockexchange.com **pinnacle:**pinnacledata2.com
apex:apex.com.tw **joinquant:**joinquant.com

Table 2.11: Summary of publications

We further examined the evaluation metrics used by the reviewed works. Recall that Section 2.4.1 presented the different financial and machine learning evaluation metrics observed by our review. As Table 2.12 shows, returns constitute the most commonly used comparison measure for obvious reasons, especially for trade strategy and price prediction; the most common objective is profit maximization. It is also common to see different derivations of returns across different time horizons, including daily, weekly, and annual returns (Théate and Ernst, 2020; Jingyuan Wang et al., 2019; C. Zhang et al., 2020).

Although machine learning metrics such as accuracy and MSE are typically combined with financial metrics, it is expected that the primary focus remains on financial metrics; hence, these are the most commonly observed. The following observations can be made based on the quantified evaluation metrics presented in Table 2.12:

- Returns is the most common financial evaluation metric because it can more intuitively evaluate profitability.
- Maximum drawdown and Sharpe ratio are also common, especially for trade strategy and price prediction specialization.
- The Sortino and Calmar ratios are not as common, but they are useful, especially given the Sortino ratio improves upon the Sharp ratio, and the Calmar ratio adds metrics related to risk assessment. Furthermore, neither is computationally expensive.

TS: Trade Strategy, PP: Price Prediction, MS: Market Simulation, SS: Stock Selection
PM: Portfolio Management, RM: Risk Management, HS: Hedging Strategy

	TS	PP	MS	SS	PM	RM	HS
Returns	13	8	2	2	4	1	0
MDD	8	2	1	2	0	0	0
Sharpe Ratio	7	3	1	1	3	0	0
Sortino Ratio	3	0	0	0	0	0	0
Calmar Ratio	3	0	0	0	0	0	0
Accuracy	3	1	0	2	0	0	0
Volatility	3	0	0	0	0	0	0
Recall	2	1	1	1	0	0	0
Precision	2	2	1	1	0	0	0
F-score	2	1	1	1	0	0	0
VaR Threshold	0	0	0	0	0	1	0
MAE	1	1	0	0	0	0	0
MAPE	1	1	0	0	0	0	0
MSE	1	3	0	0	0	0	1

Table 2.12: Quantifying evaluation measures used in different specializations

- For completeness, some studies include machine learning evaluation metrics such as accuracy and precision; however, financial evaluation metrics remain the focus when backtesting.
- Mean square error is the more common error type used (i.e., more common than MAE or MAPE).

Finding: trade strategy

A good understanding of the current and historical market state is expected before making buying and selling decisions. Therefore, it is understandable that DRL is particularly popular for trade strategy, especially in combination with LSTM. The feasibility of using DRL for stock market applications is addressed in (Y. Li et al., 2020), which also articulates the credibility of using it for strategic decision-making. That paper compares implementations of three different DRL algorithms with the Adaboost ensemble-based algorithm, suggesting that better performance is achieved by using Adaboost in a hybrid approach with DRL.

The authors of (Jingyuan Wang et al., 2019) address challenges in quantitative financing related to balancing risks, the interrelationship between assets, and the interpretability of strategies. They propose a DRL model called *AlphaStock* that uses LSTM for state

management to address the issue. For the interrelationship amongst assets, they used a Cross-Asset Attention Network (caan) using an Attention Network. This research uses the buy-winners-and-sell-losers (BWSL) trading strategy and is optimized on the Sharpe ratio, evaluating performance according to profit and risk. The approach demonstrates good performance for commutative wealth, performing over three times better than the market. Although there could be some questions regarding the way the training and test sets were divided, especially given cross-validation was not used, this work demonstrates an excellent implementation of a deep learning architecture using stock market data.

Elsewhere, (Théate and Ernst, 2020) maximizes the Sharpe ratio using a state-of-the-art DRL architecture called the trading deep Q-Network (TQDN) and also proposes a performance assessment methodology. To differentiate from the deep Q-network (DQN), which uses a CNN algorithm as the base, the TQDN uses an FFNN along with certain hyperparameter changes. This is compared with common baseline strategies, such as buy-and-hold, sell-and-hold, trend with moving average, and reversion with moving average, producing the conclusion that there is some room for performance improvements. Meanwhile, (Z. Zhang et al., 2020) uses DRL as a trading strategy for futures contracts from the Continuously Linked Commodities (CLC) database for 2019. Fifty futures are investigated to understand how performance varies across different classes of commodities and equities. The model is trained specifically for the output trading position, with the objective function of maximizing wealth. While the literature also includes forex and other kinds of assets, we focused on stock/equities. Other DRL applications include (Chakole and Kurhekar, 2020), which combines DRL with FFNN, and (J. Wu et al., 2019), which combines DRL with LSTM.

Among non-DRL architectures, the most common we observed were CNN and LSTM. In (G. Hu et al., 2018), Candlestick charts are used as input for a CAE, primarily to capture non-linear stock dynamics and long periods of historical data are represented as charts. The algorithm starts by clustering stocks by sector and selects top stocks based on returns within each cluster. This procedure outperforms the FTSE 100 index over 2,000 backtested trading days. It would be interesting to observe how this compares to using the numbers directly instead of using the chart representation.

Given moving average convergence/divergence (MACD) is known to perform worse than expected in a stable market, (Lei et al., 2020) uses *residual network (ResNet)*, a layer-skipping mechanism, to improve its effectiveness. The authors propose a strategy called MACD-KURT, which is based on *ResNet* as an algorithm and Kurtosis as a prediction target. Meanwhile, (Y. Y. Chen et al., 2018) uses a filter bank to generate 2D visualizations using historical time series data. Fed into CNN for pair trading strategy, this helps to improve accuracy and profitability. It is also common to observe LSTM-based strategies,

either for converting futures into options (J. M. T. Wu et al., 2020), in combination with Autoencoders for training market data (Koshiyama et al., 2020), or in more general trade strategy applications (Chalvatzis and Hristu-Varsakelis, 2020; Silva et al., 2020; T. Sun et al., 2019; Jun Wang et al., 2020).

Finding: price prediction

The *Random Walk Hypothesis*, popularized by (Malkiel, 2023), suggests that stock price changes in random ways, similar to a coin toss, precluding prediction. However, because price changes are influenced by factors other than historical price, numerous papers and practical applications combine all of these to attempt to obtain some insight into price movement. Given the temporal nature of buying and selling, the price prediction specialization also requires some degree of historical context. For this reason, RNN and LSTM are, unsurprisingly, often relied on. However, what is surprising is the novel use of CNN for this purpose, either as an independent algorithm or in combination with RNN algorithms.

For example, (Jia Wang et al., 2019) takes inspiration from RNN applications involving observing repeating patterns in speech and video, proposing a convolutional LSTM-based Variational Sequence-to-Sequence model with Attention (CLVSA) as a hybrid comprising RNN and convoluted RNN. The paper also introduces Kullback-Leibler divergence (KLD) to address overfitting in financial data. This work follows an optimal backtesting method involving training and testing in a sliding windows approach for eight years. Specifically, from the start of the period, the model is trained on three years of data, evaluated on one week of data, and tested the following week. Then, the training regimen shifts forward by a week before being repeated until the end of the period. However, there is no indication of whether the model is updated (i.e., online learning) or a net-new model is introduced for each sliding window. The latter is suspected. Nonetheless, the experiment shows that the algorithm produces very high returns. Elsewhere, (Baek and Kim, 2018) proposes an LSTM architecture called *ModAugNet* as a data augmentation approach designed to prevent overfitting of stock market data.

Although most algorithms use data from market trades, DeepLOB (Z. Zhang et al., 2019) uses limit order book (LOB) data with Google's Inception Module CNN to infer local interaction and feed the output to an LSTM model. It uses a CNN filter to capture spatial structure in LOB and LSTM to capture time dependencies, achieving Accuracy/Precision/Recall/F1 in the 60–70% range. The study also performs a minor simulation to test a mock trade strategy using the model's prediction. It would be interesting to see results on returns based on a full trade strategy or portfolio management.

In another use of LSTM with other architectures, (Zhao et al., 2018) incorporates fundamental and technical indicators to create a market attention model featuring a temporal component to learn a representation of the stock market. They propose *MarketSegNet*, a convolutional Autoencoder architecture that uses an image of numerical daily market activities to generate a generic market feature representation. The generated features are subsequently fed into an LSTM architecture to generate the prediction model. The results of such an approach compared with a model using actual numbers would be interesting to consider. Elsewhere, (H. Zhang, Q. Liang, Siheng Li, et al., 2020) compares LSTM with two different LSTM hybrid architectures, one using Autoencoder and one using CNN. Although the hybrid versions demonstrate better performance on accuracy tests, one hybrid's performance is only slightly better than non-hybridized LSTM in terms of Returns/Sharpe Ratio. Meanwhile, (Fang et al., 2019), combines a non-NN Regression model with LSTM, concluding that the hybrid is better than the plain LSTM in terms of accuracy but less stable when backtested.

In terms of non-LSTM architectures, (J. Wang et al., 2018) uses a one-dimensional CNN for price prediction, with the results suggesting that the model can extract more generalized feature information than traditional algorithms. This claims to be the first application of CNN on financial data, with the authors suggesting that their method achieves a significantly higher Sharpe ratio than Support Vector Machines (support vector machines (SVM)), FFNN, and simple buy-and-hold. Furthermore, the work proposes a weighted F-score that assigns priority to the different errors based on how critical they are. It is suggested that the weighted F-score works better than the traditional F-score for financial data. Finally, (H. Zhang, Q. Liang, R. Wang, et al., 2020) achieves a promising performance with a much simpler approach, using an Autoencoder algorithm for feature extraction alone.

Finding: portfolio management

Portfolio management represents another specialization area that relies heavily on DRL. In (Z. Liang et al., 2018), three state-of-the-art gameplay and robotics DRL algorithms, namely, Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), and Policy Gradient (PG) are implemented for portfolio management. The paper also proposes a new training method that improves efficiency and returns in the Chinese stock market. This approach does not produce favorable results, with the authors discovering that their model needs more data to work sufficiently in a bull market. Adjusting the objective function does not help to alleviate the risk, which is deemed too complex. However, it represents one of the earliest works to attempt to properly tackle the problem of

conducting deep learning research using stock market data.

The authors of (Park et al., 2020) also use DRL—specifically, Q-Learning—for optimal portfolio management across multiple assets. Departing from a formulated trading process, they use an MDP in which the action space, with respect to size, is the trading direction. They also use a mapping function to derive a reasonable trading strategy in the action space and simulate actions in the space, enabling them to obtain experience beyond the available data. For a baseline comparison, the authors use known strategies, such as buy-and-hold, random selection, momentum (buy improvement in previous or sales decrease in previous, based on priority), and reversion (opposite of momentum). Their experimentation outperforms baseline comparisons in terms of overall returns.

The authors of (Guo et al., 2018) propose the Robust Log-Optimal Strategy (RLOS) as part of an ensemble of pattern-matching strategies comprising RLOS and DRL (i.e., RLOSRL) for portfolio management. This approach, based on the log optimal (logarithmically optimal rate of returns), approximates the log function using Taylor expansion. It was compared with the naïve average and follows the winning strategies as a baseline. Both RLOS and RLOSRL perform better than all other approaches across multiple backtests with consistently impressive returns. Notably, the RLOSRL demonstrates superior performance, potentially significantly due to the state-aware DRL architecture. To help understand the environmental state, (Jifei Wang and L. Wang, 2019) uses FFNN with *ResNet* to address overfitting problems associated with noisy financial data, applying the strategy to regime-switching (statistical change in the data series) and concluding that *ResNet* performs better than a regular FFNN.

Finding: market simulation

Historical data are very useful and commonly used to evaluate performance over different known states. However, this features the problem that it relies entirely upon history, and the state is fully known and encapsulated into past market and economic events, introducing complications when unknown states or future what-if scenarios must be tested to ensure a robust model performance in such circumstances. Consider, for example, that a SARS-like global pandemic had been predicted for several years before the COVID-19 outbreak of 2020. It would have been useful to know how the market might react before the pandemic. In this context, market data simulation is invaluable.

The authors in (Maeda et al., 2020) propose a market DRL framework to help improve the performance of deep learning algorithms using a combination of DRL and LSTM with simulated market data. By simulating the order books for limit, market, and cancel orders,

they are able to maximize returns. This draws upon the premise that because past market actions might not represent a good indicator for the future, it is better to use simulated data for backtesting purposes. Also, specific scenarios can be created using simulated data that does not correspond to past situations, enabling the generation of data for the forecasted circumstance. This combines market simulation with trade strategy specialization. For a baseline, it compares random market actions using the same simulated data, achieving consistently impressive results.

A different approach is taken in (Raman and Leidner, 2019), which uses six weeks of real market data to generate simulated data. The authors use a DRL model to decide on a trading decision (sell, hold, or buy) for the simulated conditions, comparing the algorithm with other baseline strategies and comparing the simulated data with Monte Carlo simulations. It would be interesting to see comparisons with substantially longer time frames. Elsewhere, (Buehler et al., 2020) introduces a financial time series market simulation that relies on a very small amount of training data, using the signature of historical path segments known as "rough paths" (Boedihardjo et al., 2016; Vaswani et al., 2017) in combination with an Autoencoder. Interestingly, the authors conclude that the data generated are not significantly better than the market data and are useful for test purposes but not for real applications.

Finding: stock selection

The stock selection problem is at the core of most, if not all, stock market specializations. This represents a hard problem that some deem impossible to solve. According to (Malkiel, 2023), a group of monkeys throwing darts at a financial page will perform equally as well as experts in the task of stock selection. Nonetheless, this has not stopped researchers from exploring the problem. Although the research focus usually exceeds the singular action of selecting the stock, few studies really emphasize either this or the reasoning behind it.

The authors of (C. Zhang et al., 2020) use a feature selection technique called *DoubleEnsemble* to identify key features from stock market data. This involves training submodels (FFNN or gradient boosting ensembles (C. Zhang et al., 2020)) with weighted features to alleviate overfitting problems and stabilize them to learn with noisy financial data. To prevent stability issues and incurring huge costs by retraining models after feature removal, as traditional approaches do, a shuffling-based feature selection method has been proposed. This means that different feature sets are trained across different sample sets, and loss is measured as indicated by the missing feature. The authors backtest by hedging on a position based on model prediction, with the results showing significantly improved returns and Sharpe ratio in the context of China's A-share market. It would be interesting

to see how this compares to traditional feature reduction methods, such as Principal Component Analysis (principal component analysis (PCA)), in terms of performance, compute cost, and returns.

More sophisticated architectures have also been used. For instance, (J. Yang et al., 2019) uses CNN and LSTMfor a stock trading strategy based on stock selection. Their proposal builds features directly from the Chinese market, and a purchase is made from the model's prediction based on a projected profit of $>= 0.14\%$. The models perform significantly better than the baseline of CSI300 in the Chinese market, which is impressive considering transaction fees are included. Interestingly, a CNN-based architecture outperforms an LSTM-based architecture. This study features the drawback of not providing a comparison with a simple, baseline strategy, such as buy and hold.

Rather than constructing features using market data alone, (Amel-Zadeh et al., 2020) bases its predictions entirely on existing financial statement data (from Compustat), comparing RNN and LSTM with non-deep learning algorithms, such as random forest and regression. These experiments achieve a mild, slightly better-than-chance prediction rate of $53 - 59\%$, with the random forest model outperforming the deep learning algorithms in terms of returns. There is no evidence that lagged-time fundamentals are included as a factor in the feature engineering procedure.

Finding: risk management

Aiming to minimize risk to maximize returns, risk management represents an important specialization that must be incorporated into other strategies. However, our findings reveal that limited attention is focused on this specialization. Nonetheless, the recent market crash of 2020, caused primarily by the COVID-19 pandemic (Mazur et al., 2021), is likely to renew interest in this line of research, with at least one study already motivated by these events.

That study, (Arimond et al., 2020), compares FFNN, temporal CNN, and LSTM algorithms with the Hidden Markov Model (Hidden Markov model (HMM)) to estimate the VaR threshold. A VaR breach is reached when portfolio returns fall below the threshold. The model is trained to estimate the probability of regime change, referred to as regime-switching. This is commonly modeled as the change in market condition from a bull market (trending up) to a bear market (trending down). By estimating the moment of the VaR breach, it is possible to mitigate the risk to the portfolio.

Finding: hedging strategy

Similar to risk management, the hedging strategy specialization does not feature an extensive work of literature that fits our survey of backtested deep learning research in the stock market context. The authors of (Ruf and W. Wang, 2020) propose HedgeNet for generating a hedging strategy using FFNN over one period. Rather than predicting an estimate for option price and using that as the hedging strategy, a hedging ratio is predicted directly from the FFNN, the main metric of interest. This aligns with a recommendation from (Bengio, 1997).

Considering hedging strategies rely on training pairs of an asset at opposite positions, it would be interesting to see applications of state-conscious algorithms, such as DRL or LSTM, applied in this context.

2.4.4 Challenges

Previous sections have discussed what it means to conduct backtested deep learning research in the stock market context and summarized current research pursuing such a direction. Although there has been increasing focus on this area in recent years, numerous research challenges remain, as summarized below.

Availability of historical market data: At the core of studies based on stock market analysis is the availability of consistently updated historical data. Unfortunately, such data is a premium product that is not readily available, especially at high levels of granularity (i.e., intraday and tick data). Paywalls often restrict access to such data, complicating its use for academic research, especially research without significant financial backing. Institutions such as Wharton research data services (WRDS) (Wachowicz, 2020) collaborate with academic institutions to provide access to some of these kinds of data. However, the degree of access is determined by the subscription level, which depends on the importance ascribed by the subscribing institution. Nonetheless, the data remain widely inaccessible to a larger pool of institutions, making the only options either inconsistent publicly available market data or paying the premium.

Access to supplementary data: Closely related to the previous issue is access to related data types, which can be used to improve performance on modeling tasks involving financial data. Examples include fundamental data (e.g., quarterly reports) and alternative

data (e.g., news articles and tweets about the company of interest). It is important to differentiate these kinds of data because sources usually differ from those responsible for market data. Notably, Twitter (Now X) announced API access for research purposes (Tornes and Trujillo, 2021), which could help with this issue. However, there are many other kinds of potential supplementary data, and there remains some work to reach a state where such data is readily available. For example, it would be invaluable for news API services, such as `webhose.io`, to provide API access to supplementary news data for research purposes.

Long-term investment horizon: Several studies reviewed consider a relatively short investment horizon, from a few days to a few months. Given a significant amount of investments in the stock market are associated with portfolios that span decades, such as retirement funds, buying and holding *growth investment* is attractive. Growth investment expects above-average returns for young public companies, with the expectation of significant future growth. For example, Shopify (SHOP) IPO-ed at $17 in May 2015; as of Feb-2020, a share was worth ∼$530, with the price ending the year at ∼$1,100. This suggests that it was a growth investment at the early stage; identifying that character early would have produced larger-than-average returns. Such patterns could be discovered by using supplementary data as discussed. By modeling similar historical growth investments as part of an investment strategy, it might be possible to identify newer investments that can produce handsome returns for long-term investments.

Effect of capital gains tax: Several studies draw conclusions on a strategy without considering trading costs or taxation. This is more pronounced for short-term investments, for which tax rates are high (10%-37% in the US) compared to long-term investments (0%-20%). Thus, to accurately represent returns, these costs must be considered; however, this is seldom done.

Several research works conclude on a strategy without consideration for trading costs or taxation. This is more pronounced on short-term investments, where the tax rates are relatively high (10%-37% in the US), compared to long-term investments (0%-20%). It suffices to say that for an accurate representation of returns, these costs have to be put into consideration, yet it is seldom done.

Financial ML/DL framework: Many popular machine learning and deep learning frameworks, including scikit-learn (Pedregosa et al., 2011), TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2019), have improved the state-of-the-art. These frameworks are commonly used in both academic research and industrial research for production-level use

cases. Although these frameworks appeared frequently in the studies reviewed, implementations generally corresponded to the respective financial considerations, that is, we observed no real attempts to extend existing frameworks using improvements based on these specialized works.

Stock market machine learning problems involve incrementally learning using time-series data. Although this represents an online learning problem, the similarity remains to be fully appreciated. For example, ideas commonly used for concept drift in online learning research (Lu et al., 2018) appear perfectly suited to regime switch in quantitative analysis research. Meanwhile, some machine learning frameworks that are dedicated to online learning research have the tools and considerations for concept drift and prequential evaluation built into their framework. These include scikit-multiflow (Montiel, Read, et al., 2018) and River (Montiel, Halford, et al., 2020).

The absence of such frameworks for financial machine learning means that individual research teams must implement their ideas without attempting to integrate them into an open-source framework. Section 2.4 discusses protocols for machine learning research that involve proving results via backtesting. Having an accessible framework focused on deep learning research using financial data would enable the promotion of such ideas and allow research in this area to more closely conform to established industry practices. It would also enable researchers to provide specific implementations to improve the state-of-the-art, avoiding the current siloed approach that precludes real effort at cohesion.

2.5 Summary

In this chapter, we surveyed existing research work on the application of deep learning in the financial market. While numerous works of literature use stock market data for machine learning, the reader will notice that very few number works go as far as doing the due diligence of backtesting as part of their experimentation. From over 10,000 different publications that we came across at the time of our search, only 35 papers fall into this category, and we reviewed and summarized these contributions. The focus for these works falls primarily across a few areas of specialization, and we reviewed the works of literature with that in mind. We find it interesting that the works that conform to our survey were mostly published in recent years, and are mostly based on market data from the US and China.

The survey involved reviewing backtested applications of deep learning in the stock market. The backtesting requirement indicates that the research has demonstrated some

degree of due diligence, enabling consideration for real-world use. After demonstrating the nature of stock market data and common representations of these data, we summarized deep learning architectures, focusing on those used in the reviewed works of literature. This enabled the quick establishment of points of reference for discussion of the architectures in the context of those studies.

Ref.	Highlights/Pros	Problems/Cons
	Trade Strategy	
Jingyuan Wang et al., 2019	Clear implementation of DRL and LSTM with adequate historical data and extensive evaluation metrics.	Examples of interpretability should be provided for more than one timeframe.
Y. Li et al., 2020	DRL hybrid with Adaboost ensemble that provides good performance.	Discussion included machine learning evaluations that were not presented.
Théate and Ernst, 2020	Extensive evaluation criteria with adequate consideration for trading cost.	Details of backtesting not provided.
Z. Zhang et al., 2020	Includes tests across a vast amount of financial instruments and evaluation measures.	Unclear on how or why cross-validation was combined with the backtesting approach that was employed to control overfitting.
Chakole and Kurhekar, 2020	Focus on market trends using extensive financial and machine learning evaluation metrics and incorporating transaction costs	Interpretability insights needed to provide context for the good performance.
J. Wu et al., 2019	Extensive evaluation metrics and well-defined backtesting strategy.	Lacking conversation regarding interpretability.
G. Hu et al., 2018	Uses chart representations of financial data as deep learning input, producing a good performance	Numerical representation of the same data is missing, precluding a balanced comparison. Furthermore, there is no discussion of model interpretability.
Lei et al., 2020	ResNet used to improve the effectiveness of moving average indicators in terms of financial and machine learning evaluation metrics.	Minimal explanation for the model's performance.

Ref.	Highlights/Pros	Problems/Cons
Y. Y. Chen et al., 2018	Uses a significant amount of high-frequency trading data as input image in a pair trading setup using CNN.	Minimal evaluation of returns and no evaluation results presented. Furthermore, there are no comparisons with the raw numerical data.
J. M. T. Wu et al., 2020	Uses a high-frequency trading technique to predict profitability on daily options trading using LSTM. Provides extensive details regarding the backtesting approach.	No baseline comparison provided.
Koshiyama et al., 2020	Uses LSTM encoder-decoder to transfer trends across 58 different global markets with impressive results across multiple financial and machine learning evaluation metrics.	No interpretation of the model's operation or details of the featured transfer.
T. Sun et al., 2019	Predicts futures market movement using LSTM across multiple criteria, including simulated live trading. Additionally, multiple well backtested models are generated with different parameters and time windows.	Lack of clarity regarding why the presented model's accuracy is worse than chance. A baseline comparison with financial metrics could provide that clarity.
Silva et al., 2020	Clear presentation of the strategy, and evaluation across multiple financial and machine learning criteria.	No insights or explanations regarding the output of the LSTM model employed.
Jun Wang et al., 2020	Combines LSTM with market indicators in a novel manner with promising results.	The presented evaluation is unclear, and no insights are offered regarding the model's performance.
Chalvatzis and Hristu-Varsakelis, 2020	Nine deep learning and machine learning models are combined with LSTM in an ensemble; a well-formalized trading strategy, training, and testing conducted using a practical rolling windows approach and a complete set of evaluation criteria.	Although the general discussion regarding evaluation is extensive, it does not provide insights into the model's performance in relation to the input features.
Price Prediction		

Ref.	Highlights/Pros	Problems/Cons
Jia Wang et al., 2019	Convolutional LSTM enables price prediction with improved performance while controlling for overfitting.	Lack of discussion regarding explainability.
Z. Zhang et al., 2019	Combines LSTM and CNN to capture spatial structure in LOB and features sufficient backtesting.	Given the approach to backtesting, there is no indication of whether multiple models have been created or the same model is updated.
Zhao et al., 2018	Uses market charts as input for a CAE that serves as LSTM input.	Approach to backtesting unclear due to the unusual data split across training, validation, and test sets. Furthermore, lacks sufficient baseline comparisons and offers no discussion regarding model explainability.
H. Zhang, Q. Liang, Siheng Li, et al., 2020	Combines LSTM with Autoencoder and CNN for improved predictive results across financial and machine learning metrics.	Insufficient discussion regarding model explainability.
Fang et al., 2019	Regression model is combined with LSTM for better predictive performance.	Concludes that the results are not stable for backtested data.
Baek and Kim, 2018	Uses LSTM for data augmentation, specifically targeting controlling overfitting. Provides extensive results across machine learning, financial, and statistical criteria and discusses model performance.	No justification for why the work only considered price data, rendering the provided model's explanation less complete.
J. Wang et al., 2018	Uses one-dimensional CNN for price prediction, demonstrating better generalization than SVM and FFNN.	Makes an argument against a buy-and-hold baseline; however evaluation results based on the argument would be sufficient evidence. More discussion regarding model explainability needed.

Ref.	Highlights/Pros	Problems/Cons
H. Zhang, Q. Liang, R. Wang, et al., 2020	Provides good evaluation results for the use of an Autoencoder for feature reduction in an ensemble learning setup.	Lack of clarity regarding the backtesting strategy and the data splits. Furthermore, no discussion provided regarding model explainability.
	Portfolio Management	
Z. Liang et al., 2018	Early attempt at using DRL in the financial market featuring sufficient backtesting and evaluation of results.	No discussion regarding model explainability. Furthermore, the paper concludes that the results are unfavorable.
Park et al., 2020	Uses Q Learning to derive trading strategies in a simulated feature space to gain experience beyond the available data. Impressive performance in comparison to the baseline.	No discussion concerning insights into the model's decisions.
Guo et al., 2018	Ensemble of portfolio management using the existing state-of-the-art strategy with DRL to provide a vast improvement on returns.	Lack of discussion regarding model explainability, a necessity for insights into the vastly improved performance.
Jifei Wang and L. Wang, 2019	Uses ResNet to address overfitting problems when presented with noisy financial data. Sufficient backtesting results were provided across statistical and financial metrics.	No insights into model performance were provided to help understand the data features contributing to the performance.
	Market Simulation	
Maeda et al., 2020	Uses DRL and LSTM to simulate market data, enabling the creation of theoretical market conditions with impressive results for returns compared to the baseline.	Lack of discussion regarding model explainability.
Buehler et al., 2020	Provides a good overview of the theories involved in generative financial data modeling.	Although there is an argument that no value is derived from using more data, it is worth investigating including a comparison with more kinds of real data based on multiple market scenarios.

Ref.	Highlights/Pros	Problems/Cons
Raman and Leidner, 2019	Simulates up to a year of market data using only six weeks of real market data; simulated data is used with DRL for test trading decisions with sufficient baseline comparisons.	No financial evaluation metrics for the simulated trades. Furthermore, performance implications of a longer time frame for input and simulated data would be useful.
	Stock Selection	
C. Zhang et al., 2020	Combines LSTM with boosting ensembles to identify key market features and reduce overfitting.	No comparisons with traditional feature reduction methods such as PCA and no discussion of model explainability.
Amel-Zadeh et al., 2020	Using only fundamental data, compares RNN and FFNN models with non-deep learning algorithms; the non-deep learning methods outperform the deep learning models.	No information on the completeness of the input data, namely, lagged dates, and no insights into the model's performance.
J. Yang et al., 2019	Compares CNN withLSTM, with features derived from profit indicators.	No comparisons with other baseline strategies and no discussion of model explainability.
	Risk Management	
Arimond et al., 2020	Specifically targeted at using CNN and LSTM to estimate VaR with a focus on future research potentials.	Fails to formally present the baseline evaluation results or provide an explanation or suggestions regarding potential model performance.
	Hedging Strategy	
Ruf and W. Wang, 2020	Uses FFNN to predict a derived metric that it uses in a hedging strategy with promising results.	No consideration of the state of the model or discussion of insights from the model output.

Table 2.13: Highlights of and problems with publications reviewed

Table 2.13 presents the highlights of concerns with the studies reviewed, demonstrating that while they represent impressive work in different capacities, many insufficiently discuss model explainability, and none focus on the long-term investment horizon. Also, while

these works mostly combine market and fundamental data, it is still difficult to include alternative data, such as news texts or Twitter data, which can enrich the modeling process. This is largely due to the unavailability of such data, especially for long historical time windows. As this area of research continues to mature, we trust that more attention will be paid to these issues and that research interests can influence the industry at large to make most of the cost-prohibitive data forms available for research purposes.

In the next chapters, we introduce our novel temporal Transformer architectures, that address concerns identified with incorporating historical time windows and explainability.

Chapter 3

Embedding historically similar trends in temporal Transformers

This chapter represents the work as presented in two of our research papers, namely:

- *Similarity embedded temporal Transformers: Enhancing stock predictions with historically similar trends* was presented at the 26th International Symposium on Methodologies for Intelligent Systems (ISMIS 2022), and published in the proceedings (Olorunnimbe and Viktor, 2022) (Olorunnimbe and Viktor, 2022b). This paper won the Best Student Paper Award. An extended version of this paper titled "*Towards efficient similarity embedded temporal Transformers via extended timeframe analysis*" has been accepted for publication in the Complex & Intelligent Systems journal (Olorunnimbe and Viktor, 2024) (Olorunnimbe and Viktor, 2024c).
- In this chapter, we introduce two models for multi-horizon forecast called SeTT and r-SeTT, which factor in the heteroscedasticity of financial market time series into temporal Transformer. We conduct extensive experimentation across multiple timeframes using different lengths of time and our experimental evaluations confirm that the models outperform both the classical financial models and the baseline temporal Transformer architecture in terms of predictive performance.

3.1 Overview

The use of historical time series data to make predictions about the prices of financial instruments such as stocks, bonds, and futures is an important area of research and represents

an integral component of automated trading and portfolio management systems (Olorunnimbe and Viktor, 2022a; Sezer et al., 2020). In recent years, deep learning approaches have outperformed classical machine learning methods in a variety of tasks, including financial applications. The introduction of the Transformer architecture (Vaswani et al., 2017) has been pivotal in this advancement, by processing sequential data more effectively than traditional deep learning methods such as RNN and LSTM. Building upon this, the temporal Transformer architecture further tailors the architecture for time-series analysis (Lim, Arik, et al., 2021).

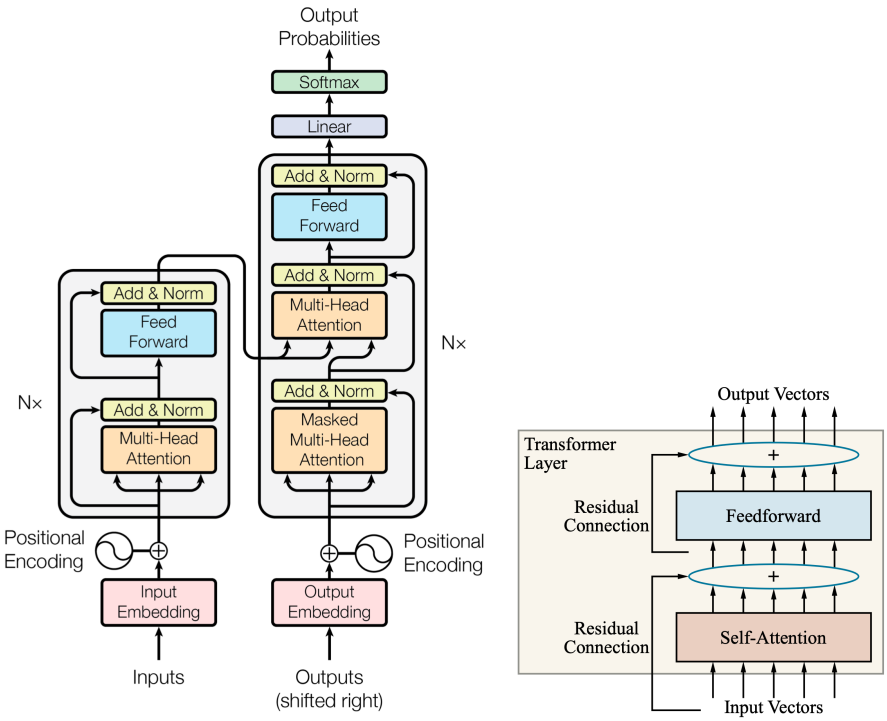
However, temporal Transformers do not consider financial market data. That is, current temporal Transformer architectures do not fully address the non-constant nature of the volatility of data across different periods of time. They also do not utilize the fact that the volatility of a specific day is strongly related to the previous day (Prado, 2018). We build on the state-of-the-art temporal Transformer architecture to develop a deep learning technique that is aware non-constant nature of financial time series in the face of the volatility of the financial market. We start by explaining what the Transformer model and attention mechanism are, before presenting our model architectures.

3.2 Attention mechanism and Transformers

Recall that in Section 2.3.1, we described RNN as a neural network that keeps a sequential representation of previously seen data as a hidden state. We also explained that it has the issue of not retaining information over a long sequence of data; an issue that while its derived versions such as LSTM and GRU were introduced to solve, still exist for the application of recurrent models over a very long-range. In this section, we give a brief explanation of the Transformer model and its underlying self-attention that was developed to solve some of these issues.

3.2.1 Transformer

The domain of deep learning in sequential modeling, particularly in NLP, has traditionally relied on RNN and its different variants, such as LSTM. RNN maintains information across sequences using an internal state that gets updated recursively, acting as a summary of past data points (Lim and Zohren, 2021). However, this method suffers from several limitations, such as the vanishing gradient problem and difficulty capturing long-range dependencies in sequences (Goodfellow et al., 2016). To address these limitations, Vaswani



(a) A full Transformer architecture showing encoder and decoder (Russell and Norvig, 2021)

(b) A single layer Transformer encoder (Vaswani et al., 2017)

Figure 3.1: Illustration of the Transformer architecture.

et al. (Vaswani et al., 2017) introduce the Transformer architecture, a novel attention-based model that has since become the standard architecture for sequential deep learning. The Transformer's success lies in its use of *attention mechanisms*, which provide improved parallelism and overcome the limitations of traditional RNNs. This also simplifies the complexity of sequential operations done by the attention mechanism from $Olog(n)$ to $Olog(1)$ when compared to RNN. The attention mechanism is formulated from the query (Q), key (K), and value (V) matrices comprising q, k, and v vectors, all generated from the input sequence. The q vector represents the information being queried, the k vector represents the information being compared against, and the v vector represents the query result to be generated (Russell and Norvig, 2021). The attention weights are calculated by taking the dot product of the Q and K matrices divided by the dimension of the k vectors (d_k), before normalizing the resulting scores using softmax. These attention weights are then used to compute weighted values of V that represent the attention mechanism (Vaswani

et al., 2017).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

where (Russell and Norvig, 2021):

- $Q = q_i = W_q x_i$: the query vector been attended from, similar to a search query
- $K = k_i = W_k x_i$: the key vector been attended to, similar to keys in a dictionary
- $V = v_i = W_q x_i$: the values vector to be generated
- $A(QKV) = c_i = \sum_j \text{softmax}(\frac{q_i.k_j}{\sqrt{d_k}}).v_j$: sum of all previous positions in sequence. Multiple instances of this, known as *multiheaded attention* are average to reduce potential loss of information.

.

The q, k, and v vectors are computed by passing the input sequence through a linear transformation to learn weights matrices W_q, W_k, & W_v using a FFNN for the final output. The same weights, W_q, W_k, & W_v, are shared across all sequence elements, allowing for efficient computation of attention scores and representation of entire sequences in parallel. This also enables the Transformer architecture to process input sequences of varying lengths. In the encoder component of the Transformer architecture, input sequences are associated with the order in which they occur in the data sequence using position encoding made of sinusoidal sine and cosine functions. Sinusoidal functions are used to enable the model to extrapolate to any arbitrary sequence length, beyond the ones it was trained on (Vaswani et al., 2017). The output sequence is subsequently generated from the decoder, with the encoded representation of the encoder as input.

Transformer-based models have emerged as state-of-the-art in deep learning approaches. The development of Transformer-based large language models (LLM) such as BERT (Wilkinson et al., 2016) and InstructGPT (Ouyang et al., 2022) has propelled the advancement of artificial intelligence and its prevalence in public discourse. Transformer models are increasingly applied in the financial domain, and they have demonstrated the potential to outperform traditional statistical methods (Lim, Arik, et al., 2021).

More research has been conducted to apply all these learnings from NLP to time series data (Shiyang Li et al., 2019; Lim, Arik, et al., 2021). This thesis goes one step further by applying the Transformer model to financial time series while taking into consideration the unique characteristics of financial data.

3.2.2 Temporal Transformer

The Transformer model was designed for non-temporal tasks such as NLP and is not well-suited for handling sequential data with temporal characteristics, such as speech or time series data. Positional encoding is insufficient to capture temporal relationships or local dependencies between elements in a sequence since it uses a fixed positional representation of each element. Temporal Transformers with locality awareness using learnable positional encoding were developed to solve this issue (Shiyang Li et al., 2019; Lim, Arik, et al., 2021). Furthermore, despite the inherent interpretability of the Transformer model stemming from its attention mechanism (Vaswani et al., 2017), it cannot distinguish the relative importance of features across different time steps.

Learnable positional encoding enables the encoding of temporal relationships between local data points; this information is used to search through the attention network. It extends the standard multi-head attention mechanism to account for temporal relationships between data points at relatively short proximity. In temporal fusion Transformer (TFT), an LSTM sequence-to-sequence layer is used for local processing, focusing on data recency and establishing short-term relationships within the encoder and decoder, while the self-attention network is used for long-term dependencies (Lim, Arik, et al., 2021). All of the past information within a finite lookback (current) window is incorporated into each sequence element using LSTM.

Let ξ_t represent the value derived from the original sequence element, X_t, after a non-linear transformation at each time t. The q, k, & v vectors are not derived from ξ_t, but rather from the output of the sequence-to-sequence layer, $\phi(t, n)$, with locality-aware temporal features in a procedure called local processing (Lim, Arik, et al., 2021). $\xi_{t-l:t}$ is fed into the LSTM encoder and $\xi_{t+1:H}$ is fed into the decoder, generating context vectors as a uniform temporal feature set, $\phi(t, n) \in \{\phi(t, -l), ..., \phi(t, H)\}$ for each timestep. n is the position index of t in the sequence, l is the lookback window size, and H is the forecast horizon. Local processing captures local temporal patterns and relationships between adjacent time steps via the sequence-to-sequence layer, providing useful local features before longer-term dependencies are modeled in the self-attention layer.

In a time series scenario, the key vectors are generated from the historical data (e.g. the

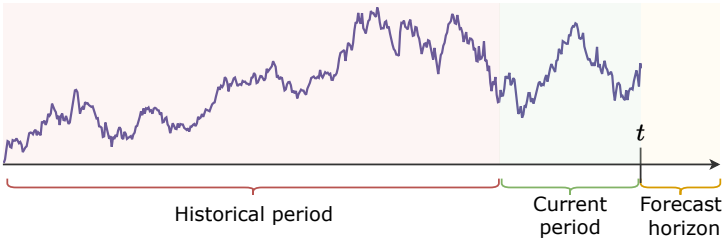


Figure 3.2: Financial time series showing different timeframes

last 2 years) and they represent the data that the model is looking at. The query vectors are generated from the input data (e.g. the last 30 days), representing the questions the model is asking about the historical data. The value vectors represent the answers to those questions (i.e. the prediction horizon). The attention mechanism compares the query vectors to the key vectors and determines how closely they match. The closer the match, the more attention the model pays to that particular data point. The final output of the attention mechanism is a weighted combination of the value vectors, based on the most relevant historical data points. Multi-head attention allows the decoder to search for the most relevant part of the encoder sequence based on the weights learned in the training process.

The temporal Transformer architecture was shown to perform well on various sequential data tasks, including electricity, traffic, and financial volatility forecasts (Lim, Arik, et al., 2021). It captures the interpretability of different time steps and analyses global temporal dynamics using shared weights for the Q, K, & V matrices in each attention head, aggregated across all heads. However, no strong, persistent patterns were observed for the financial data under experimentation because of the "high degree of randomness". We posit that this is because of the conditional heteroscedasticity of financial time series data.

This study extends the TFT architecture, focusing on financial time series with consideration for financial data characteristics. To assess the efficacy of our models, we conduct extensive experimentation via hyperparameter optimization and extended timeframe analysis. We aim to optimize their performance to provide a fair comparison between models. This is particularly important when working with complex algorithms and unique data (L. Yang and Shami, 2020). It allows us to accurately assess the strengths and weaknesses of each model and make informed decisions across different timeframes. In the next section, we provide details of our SeTT architectures (Olorunnimbe and Viktor, 2022b), which employ the notion of similarity vectors to make the temporal Transformer aware of the

distributions of domain-specific time series.

3.3 Incorporating financial properties into attention mechanism

Building on the contextual representation of the Transformer model and the importance of locality for time series as explained in the sections 3.2.1 & 3.2.2, we further explore historical precedence from financial time series evaluation. We make use of the look-back window of a financial data sequence to provide contextual representation to a deep learning architecture for financial time series data. In Section 3.2.1, we explained the self-attention, and how the query is used with the key to query the attention network in a Transformer model, across the timewise sequence. In financial data such as the stock market time series, the sequence could span through a very long period, with some similarities across different historical periods.

Considering the default representation of the attention mechanism assumes IID, this opens up an opportunity to aid it in the search across the time-space for domain-specific data sequences such as this.

3.3.1 Similarity embedded temporal Transformers

Building on the importance of locality for time series, we explore historical precedence in financial time series evaluation. Specifically, we make use of the lookback window to provide a contextual representation of a deep learning architecture for financial time series data. A financial time series is modeled as finite historical targets $y_{t,t\in\{1,\ldots,T\}}$, where y_t is the target at the current point in time, $y_{\tau,\tau\in\{1,\ldots,H\}}$ is the forecast horizon, and $y_l \leftarrow y_{i,i\in\{T-l+1,\ldots,T\}}$ are the targets of the most recent time series of size l (i.e. the current window). Our SeTT architecture introduces the concept of a sliding window to learn similar time series.

Our similarity embedded temporal Transformer (SeTT) algorithm consists of four steps. (1) A current window and multiple historical windows are generated from the time series data. (2) The historical windows are individually compared with the current window. These comparisons return vectors of $1s$ if similar or $0s$ if dissimilar, each of size l. (3) The individual vectors generated by the comparison are combined into a similarity vector. (4) The similarity vector is embedded into the temporal Transformer architecture during training to mute the dissimilar time series. Algorithm 3.1 depicts the pseudocode for the SeTT algorithm.

Algorithm 3.1 similarity embedded temporal Transformer (SeTT)

Input:

$X_{t,t\in\{1,\dots,T\}}$ ▷ time series features of size T

$y_{t,t\in\{1,\dots,T\}}$ ▷ time series targets (Δ`price`) of size T

l ▷ current window size; $l < T$

Output: $\hat{y} \sim y_{\tau,\tau\in\{1,\dots,H\}}$ ▷ vector of H future quantile forecasts

$SimVec \leftarrow$ GenerateSimVec(y, T, l)

$\phi(t, n) \sim X$ ▷ temporal features generated from X

$Q \leftarrow W_{q\phi}$, $K \leftarrow W_{k\phi}$, $V \leftarrow W_{q\phi}$ ▷ initialise attention

$\mathcal{L}(y, W) = L_q(y, \hat{y}, q)$ ▷ update W_q, W_k, W_v by minimising quantile loss

repeat

A(Q, K, V) $\leftarrow$ softmax $\left(\frac{Q\{K \cdot SimVec\}^T}{\sqrt{d_k}}\right) V \cdot SimVec$

$\hat{y} \leftarrow L_q(y, \hat{y}, q)$

until stopping condition is met

procedure GenerateSimVec(y, T, l)

$y_l \leftarrow y_{\{T-l+1,\dots,T\}}$ ▷ current window

$N \leftarrow T - l$ ▷ size of historical windows

$\mathbf{A} \leftarrow []$ ▷ initialise an array buffer

for all j in $0...N - l - 1$ **do**

$y_j \leftarrow y_{\{j,\dots,j+l\}}$ ▷ index of sliding historical windows

$c \leftarrow$ compareWin(y_l, y_j) ▷ $c \in \{0, 1\}$

$\mathbf{a} \leftarrow (\vec{1}_j, \vec{c}_l, \vec{1}_{N-l-j})$ ▷ pad both sides of l-sized vector of repeated c with $1s$

$\mathbf{A}[j] \leftarrow \mathbf{a}$

$\mathbf{h} \leftarrow \min_i\{\mathbf{A} = \mathbf{a}_{ij}\}$ ▷ column-wise minimum values of 2D-array

return $(\mathbf{h}, \vec{1}_l)$

procedure compareWin(y_l, y_s)

if $S(y_l, y_s) > \alpha$ **then return** 1 ▷ $\alpha = 0.05$; S: similarity-test function

else return 0

Historical windows

The financial market is known for its inherent unpredictability and volatility, which make the utilization of past stock prices and performance a crucial aspect of the analysis process. However, this requires some caution; according to the random walk hypothesis (Malkiel, 2023), stock price fluctuations follow a random pattern and are independent of one another,

making it unfeasible to make predictions solely based on historical movements. Furthermore, two random processes, such as stationary time series (i.e. those with non-changing mean and variance) within different fixed windows, can be considered similar if they exhibit the same statistical properties (Brockwell and R. A. Davis, 2016). This implies that although historical price movements are considered to be random, there may exist specific timeframes that exhibit similar patterns in these movements. We incorporate this concept into a temporal Transformer architecture by generating historical windows from historical price movements of a finite length to identify similar historical patterns.

This forms the foundation of our algorithm to compute the similarity vector. The choice of historical length is a trade-off–extended length may allow the model to identify long-term trends, but it also elevates the probability of capturing irrelevant volatility. As a result, time series models may not be suitable for a very long historical period (R. Hyndman and Athanasopoulos, 2021). On the other hand, while a shorter timeframe may not encompass long-term patterns, it offers a more accurate portrayal of current market conditions. To derive the most advantageous results from both short-term and long-term market data, we propose to compare the representations of historical data within discrete timeframe windows with the most recent short-term data point at the present moment, y_t. The historical windows comprise of all the available windows $y_N \leftarrow y_{i,i\in\{1,\dots,T-l\}}$ of length l, using a stride of one (1). Exploring additional strides is outside the specific focus of this research, but may offer different perspectives and will be considered in future explorations. In the first variant of our SeTT algorithm, these windows are generated by sliding a vector of size l across the historical timeframe of length N using the time series target values, y.

Financial markets often exhibit a high degree of volatility, characterized by significant fluctuations in stock prices. This volatility can result in frequent uninterrupted sequences of either positive or negative price changes, called runs, which have been shown to have a significant impact on the trading landscape (Prado, 2018). Research suggests that investors consider stocks with shorter up-and-down movements to be less risky than those with longer run-length, leading to further bias in price movement (Raghubir and Das, 2010). To consider runs, we introduce the second variant of our SeTT algorithm, called r-SeTT. In r-SeTT, we incorporate the binary representation of the time series targets (Δ`price`) within the specified time horizon. That is,

$$y_r = \textsc{Run}(y_i) = \begin{cases} 1 & \text{if } y_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where y_i is a price change and y_{ri} is the binary representation of the price change. For the new variant, we replace y in Algorithm 3.1 with y_r in Equation 3.2 before comparison.

The historical run windows at y_t are all the available positive (1) or negative (0) price change windows of length l, $y_{rN} \leftarrow y_{ri,i\in\{1,\dots,T-l\}}$, not in the current run window $y_{rl} \leftarrow y_{ri,i\in\{T-l+1,\dots,T\}}$. We believe that run is an indication of a changing market regime and will be reflected in how historical windows compare with the most recent timeframe.

Comparison between windows

After the historical windows are generated, they are individually compared to the current window. To test for similarity on the SeTT algorithm, we employ Cramér–von Mises (CM), Kolmogorov–Smirnov (KS), and Epps–Singleton (ES) tests of fit. These are non-parametric tests for the null hypothesis (H_0) that two independent samples have the same probability (Erlemann et al., 2022; Goerg and Kaiser, 2009; Hollander et al., 2013). H_0 in all tests is that two samples are drawn from the same distribution.

Theorem 3.1 (Two-sample Cramer-von Mises test)**.** *Suppose two independent observations $X^m \to \mathbb{R}^m = x_1, x_2, \dots, x_m$ and $X^n \to \mathbb{R}^n = x_1, x_2, \dots, x_n$ with the empirical distribution functions $F_m(x)$ and $G_n(x)$ and the combined empirical distribution of the entire observation, $H_{m+n}(x)$, defined as:*

$$F_m(x) = \frac{1}{m}\sum_{i=1}^{m}(X_i \leq x);\; G_n(x) = \frac{1}{n}\sum_{i=1}^{n}(X_i \leq x) \tag{3.3}$$

$$H_{m+n}(x) = \frac{mF_m(x) + nG_n(x)}{m+n} \tag{3.4}$$

The Cramer-von Mises statistical test for the null hypothesis that X^m & X^n are independent and identically distributed (IID) is given by (Erlemann et al., 2022):

$$\textit{CM test statistic} = \frac{mn}{m+n}\int_{-\infty}^{\infty}\{F_m(x) - G_n(x)\}^2 n H_{m+n}(x) \tag{3.5}$$

Theorem 3.2 (Two-sample Kolmogorov-Smirnov test)**.** *Let X^m & X^n be the same two independent observations with the same empirical distribution function. The Kolmogorov-Smirnov statistical test evaluates the null hypothesis that the two samples are IID is given by (Hollander et al., 2013):*

$$\textit{KS test statistic} = \max|F_m(x) - G_n(x)| \tag{3.6}$$

Theorem 3.3 (Two-sample Epps-Singleton test)**.** *Let X^m & X^n be the same two independent observations. The empirical characteristic function of each sample is defined as:*

$$\varphi_m(x) = \frac{1}{m}\sum_{i=1}^{m} e^{\sqrt{-1}xX_i};\ \varphi_n(x) = \frac{1}{n}\sum_{i=1}^{n} e^{\sqrt{-1}xX_i} \tag{3.7}$$

The Epps-Singleton statistical test to evaluate the null hypothesis that X^m & X^n are IID is given by (Erlemann et al., 2022):

$$ES\ test\ statistic = \max|\varphi_m(x) - \varphi_n(x)| \tag{3.8}$$

Corollary 3.1 (Compare windows for similarity)**.** *Let l be the fixed size of the current window, y_l. The same size is used for each of the sliding windows, y_i, in the historical data. The p-value is computed based on the values in the distributions to be compared, and the null hypothesis is rejected if the p-value is less than 5% ($\alpha = 0.05$). Otherwise, it is accepted. A consistent sequence of similar historical windows is determined from the historical windows. Windows similar to y_l are represented as an l-sized vector of $1s$ ($\vec{1}_l$), whereas dissimilar windows are represented by an l-sized vector of $0s$ ($\vec{0}_l$). That is,*

$$\vec{c_{il}} = \begin{cases} \vec{1}_l & if\ \textsc{compareWin}(y_l, y_j) \geq \alpha \\ \vec{0}_l & otherwise \end{cases} \tag{3.9}$$

Since the input to the r-SeTT similarity vector routine is a binary representation, we use the *Hamming distance (hd)* to measure the similarity between windows. The Hamming distance captures the positions of differences between paired binary windows (Leskovec et al., 2020), thereby determining the similarity of different run windows.

$$D_H(y_l, y) = \sum_{i=1} |y_{il} - y_i| \tag{3.10}$$

where y_{il} and y_i are the ith elements of the windows been compared. We used 0.7 as the threshold for the minimum proportion of disagreement, meaning that two windows are considered dissimilar if less than 70% of the paired elements are the same. This is consistent with current knowledge on thresholded Hamming distance search to represent the lowest value of the minimum recall rate (Ong and Bober, 2016), and it is further validated by our preliminary inspection.

Similarity vector

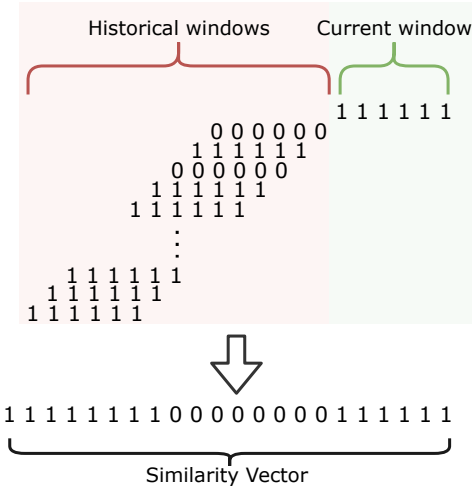


Figure 3.3: An illustration of a similarity vector of current and historical windows for a stock symbol.

A stock symbol is a set of letters representing a publicly traded company or asset on the stock market. For each stock symbol, a similarity vector of length T is generated from the vectors produced by comparing the historical windows with the current window. In Algorithm 3.1, we defined T as the total size of the time series, l as the current window size, and $N = T - l$ as the size of the historical window.

Corollary 3.2 (Similarity vector)**.** *Let $M = N - l + 1$. c_i is the binary result of the comparison between the current window and the window at $i \in \{1, ..., M\}$. A vector of binary values is first computed from the historical windows as follows:*

$$\mathbf{A}^{MxN} = \{\mathbf{a}_i \in \{0,1\}^N, i = 1, ..., M\} \tag{3.11}$$

$$= [\mathbf{a}_1, ..., \mathbf{a}_M] \tag{3.12}$$

$$\textit{where } \mathbf{a}_i = (\vec{1}_{i-1}, \vec{c}_{il}, \vec{1}_{M-i}) \mid \forall i \in \{1, ..., M\} \tag{3.13}$$

$$\mathbf{h} = \min_i \{\mathbf{A}^{MxN} = \mathbf{a}_{ij}\} \tag{3.14}$$

The similarity vector is made up of a binary representation of the current window, depicted as a vector of 1s with size l ($\vec{1}_l$) appended to the binary representation of the historical windows, $\mathbf{h}$.

$$\mathbf{s} = (\mathbf{h}, \vec{1}_l) \tag{3.15}$$

A series of $1s$ represents a consistent similarity with the current window, potentially across different time horizons; 0s indicate dissimilarity. The current window is also part of the similarity vector, instantiated as a series of $1s$. The vectors are then collapsed using column-wise matrix multiplication to give a vector of the same length as T, as illustrated in Figure 3.3. A matrix consisting of individual vectors for n stock symbols, $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n]$, is incorporated into the model architecture.

Model architecture

As the system overview in Figure 3.4 illustrates, the attention mechanism employs the similarity vector when searching through the temporal attention network. The same similarity vector is used for the entirety of the training period, comprising a constant complexity $O(1)$ addition to the regular temporal Transformer architecture. This ensures that the comparative efficiency of the Transformer model is retained (Tay et al., 2022). The Transformer architecture searches the attention mechanism for the most similar encoder sequence in the attention network, assuming that the input data is homoscedastic, with constant mean and variance (Vaswani et al., 2017). In this study, we ensure that during training and inference, the model only searches through historical windows in the attention network that are similar to the current window by muting dissimilar sequences using the similarity vector.

The temporal multi-head attention mechanism captures long-term dependencies based on discrete local processing from the LSTM units. The gated residual network (GRN) is a neural network with gating mechanisms allowing for adaptive depth and complexity. It comprises a ResNet, giving the model the flexibility of not applying any processing when unneeded (i.e. when the data is small or noisy), adding the benefits of simplicity. By combining the outputs of local processing with the attention mechanisms, the model integrates information about both short-term and long-term dynamics. This provides a robust foundation for financial time series forecasting. For a more in-depth discussion of the temporal multi-head attention mechanism and GRN, we refer the reader to the work of Li et al. (Lim, Arik, et al., 2021), which provides a comprehensive overview of this topic.

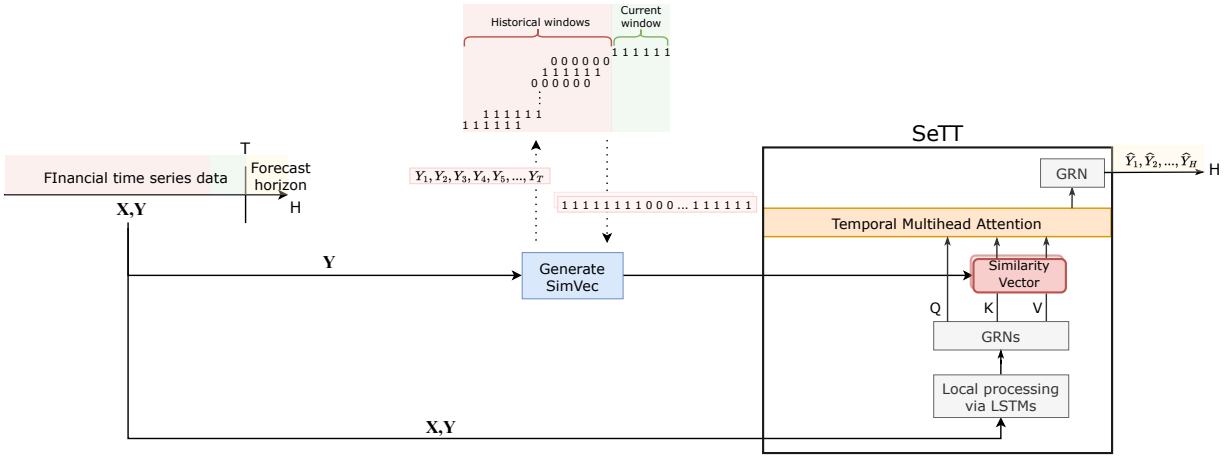


Figure 3.4: System overview showing the matrix of similarity vectors for multiple stock symbols generated from the time series data and used within the SeTT architecture. The similarity vectors are used to mute irrelevant keys and values vectors from the queries vectors of the attention mechanism.

Multi-horizon forecasting

Temporal Transformer models are designed to forecast multiple horizons and are trained using past observations combined with past and future known patterns (e.g. the day of the week). In contrast to one-step-ahead prediction, *multi-horizon forecast* (R. Wen et al., 2017) involves making price prediction multiple steps in advance, as in a trading week (5-day) prediction. This approach is more accurate and efficient than recursive one-step-ahead forecasts when the input involves dynamic and static historical and future attributes, such as financial time series (Lim, Arik, et al., 2021; R. Wen et al., 2017).

The future value of a time series is an unknown random variable in a forecast distribution (R. Hyndman and Athanasopoulos, 2021). Instead of estimating the absolute value, we estimate the probability over a range, known as a *quantile forecast* (R. Hyndman and Athanasopoulos, 2021). A quantile forecast is robust when it is difficult to make absolute predictions, as it provides probabilistic forecasts rather than assuming future distribution (R. Wen et al., 2017). This is also useful for minimizing risks associated with a financial decision, as it provides the best-case and worst-case scenarios for the prediction target over a probability range (Lim, Arik, et al., 2021). In the training process, predictions are made across all quantiles of interest. The total loss is minimized on multiple forecast horizons using *quantile loss* as the loss function (Equation 3.16).

$$L_q(y, \hat{y}, q) = \begin{cases} q(y - \hat{y}) & \text{if } y - \hat{y} \geq 0 \\ (1 - q)(\hat{y} - y) & \text{if } y - \hat{y} < 0 \end{cases} \tag{3.16}$$

$$= max(q(y - \hat{y}), (1 - q)(\hat{y} - y)) \tag{3.17}$$

For comparison with other baseline models and consistency with previous work, we used 50% (p50) and 90% (p90) quantiles (Shiyang Li et al., 2019; Lim, Arik, et al., 2021). The 50% quantile is the target forecast, as it is the median value of the predictive distribution. As an evaluation metric, we used MASE because it is resistant to outliers inherent in financial time series, and it independently scales the predictions with the actual targets of the entire history, not just the forecast horizon (R. J. Hyndman and Koehler, 2006). Other metrics such as MAE & root mean square error (RMSE) are not ideal for comparing time series models as their error magnitude is influenced by their specific evaluation scale, resulting in comparison issues, as we also observed during preliminary experiments. This issue is consistent with other time series research and is why MASE is regarded as the preferred method for evaluating the performance of time-series forecasting models (Franses, 2016; R. J. Hyndman and Koehler, 2006).

$$\text{MASE} = \frac{\frac{1}{n}\sum_{t=1}^{n} |e_t|}{\frac{1}{n-1}\sum_{t=2}^{n} |y_t - y_{t-1}|} \tag{3.18}$$

$$\tag{3.19}$$

Where e_t is the forecast error at time t, y_t is the actual value at time t, n is the length of the time series, and the denominator is the MAE of the naive forecast (i.e., the one-step-ahead forecast based on the previous observation). We used 5 days as the forecast horizon, following (Lim, Arik, et al., 2021; Olorunnimbe and Viktor, 2022b).

3.4 Experimental evaluation

In this section, we present the findings of our preliminary experimentation. We set out to evaluate the effect of the local financial window on the global awareness of the attention mechanism. We formulate the experimentation as a multi-horizon problem, allowing us to evaluate historical performance on the next forecast horizon. We validated our hypothesis

that by suppressing the attention paid to dissimilar windows during the query phase of the attention mechanism, the performance of the Transformer model performance significantly better. To confirm this, we measure our level of significance using a t-measure.

Below we explain the setup of our experimentation, including the details of the data used, and the feature engineering approach we employ to arrive at our current results.

3.4.1 Experimentation setup

Data

Industry		Symbol	Weight (%)
TECH	Information technology	AAPL, CSCO, IBM, INTC, MSFT, CRM	18.61
HEALTH	Managed health care	UNH	7.88
PHARM	Pharmaceutical, Biopharmaceutical	AMGN, JNJ	6.88
FOOD	Soft Drink, Food	KO, MCD	5.52
FIN	Financial services	V	4.34
AERO	Aerospace and defense	BA	4.sw
CONST	Construction and Mining	CAT	3.73
CONG	Conglomerate	MMM	3.38
APPAREL	Apparel	NKE	2.93
RET	Retailing	WMT	2.69
GOODS	Fast-moving consumer goods	PG	2.61
OIL	Petroleum	CVX	2.07
TELE	Telecommunication	VZ	0.97

Table 3.1: Industry and stock symbols of 20 of the 30 DJIA companies, showing their collective industry weight in the DJIA index.

We used time series stock market data and financial data to evaluate our model. We leverage the stock market symbols for the companies listed in the Dow Jones industrial average (DJIA). DJIA is a stock market index, weighted by price, consisting of 30 prominent U.S. companies across 20 industries. We require both market and corporate results data across all time frames for our experiments. Thus, we included 20 stocks (Table 3.1) in our analysis, representing 65.62% of the overall index's weight across 15 industries. These stocks were selected from the 30 available in the index because they provided complete data sets throughout the 13-year period. Stocks not included primarily either concluded before the period ended or commenced after it began.

The historical pre-processed dataset (market and fundamental) corresponding to the study's period is available for download through SimFin Analytics GmbH (SimFin)[1], an

[1] simfin.com

online financial data resource. The period represented in the data is Jan. 2007 to Nov. 2022. A limited period of 5 years, is freely available for download. Due to licensing restrictions, the full period of 13 years of data, used in this publication, is deemed proprietary data and cannot be freely disseminated within this publication. However, it is worth noting that the data is affordable, making it accessible for research purposes. This is consistent with standardized data availability specifications for reproducibility with restricted access conditions ("Reporting standards and availability of data, materials, code and protocols" 2013; *Research data policy Springer Nature* 2023).

Attribute	Abbreviation	Description	Type
Open, high, low & close prices	-	Daily open, high, low & close prices	-
Volume	-	Volume of shares traded in a day	-
Shares outstanding	-	All share available for trade	-
Earning per share (EPS)	-	Profit for each outstanding share	-
1-day log-price-change	p_logd	Log-change in closing price over 1 day	target
1-day log-volume-change	v_logd	Log-change in volume over 1 day	observed numeric
Shares turnover	s_turnover	All share available for trade	observed numeric
Price variation	p_variatn	% change in open & close prices	observed numeric
Price-Earning (P/E) ratio	per	Share price relative to EPS	observed numeric
Normalised high, low close	norm_h, norm_l, norm_c	High, low & close prices normalised with open price	observed numeric (x3)
Year, Month, Weekday	year, month, weekday	Categorical representations of year, month & weekday	known categorical (x3)
Industry	industry	Market sector of company	static categorical

Table 3.2: Features used as input to the model.

To ensure consistency during the independent trials and aid reproducibility, we used the Lakehouse medallion data design pattern of bronze, silver, and gold tables to refine the input data used in the model (Armbrust et al., 2021; Databricks, 2022). Historical daily US market data and quarterly income data from SimFin are initially compiled into bronze tables with minimal preprocessing. This raw data includes high, low, close prices, and earnings per share. From these bronze tables, we preprocessed the relevant fields to create an enriched daily share price silver table. This table includes features such as log changes, normalized prices, and PE ratios, as detailed in Table 3.2.

As mentioned, we only use 20 of the market symbols represented in the DJIA index. Full historical data is available for these symbols from Nov. 2009 to Nov. 2022, compris-

ing 759,640 records. We create a gold table consisting of just these 20 symbols, enriched with information about their DJIA industry, as that serves as relevant static metadata to the temporal Transformer model. The model reads from this table during each experiment, which is filtered in use during the evaluation period, and a monotonically increasing sequence index is added from the start to the end of each period.

Adopting an approach such as the medallion data design pattern for the feature engineering pipeline allows for easier reuse and repurposing of feature data, as specified in the Findability, Accessibility, Interoperability and Reusability (FAIR) principles of scientific data (Wilkinson et al., 2016). It also simplifies versioning, facilitating the efficient identification and correction of potential data issues and allowing for updates to be made at any step of the transformation.

Evaluation period

We constructed independent data sets consisting of information about the 20 companies shown in Table 3.1 across multiple independent extrapolation periods. To determine the periods, we used the VIX[2], a real-time index from the CBOE. The VIX measures stock market volatility and is based on the relative strength when compared with the S&P 500 index. Note that a VIX of 0–12 is considered low, 13–19 corresponds to normal, and above 20 is deemed high (T. Edwards and Preston, 2017). Higher values are expected during times of financial issues, as during the period of extreme volatility induced by COVID-19 at the start of 2020, as shown in Figure 3.5.

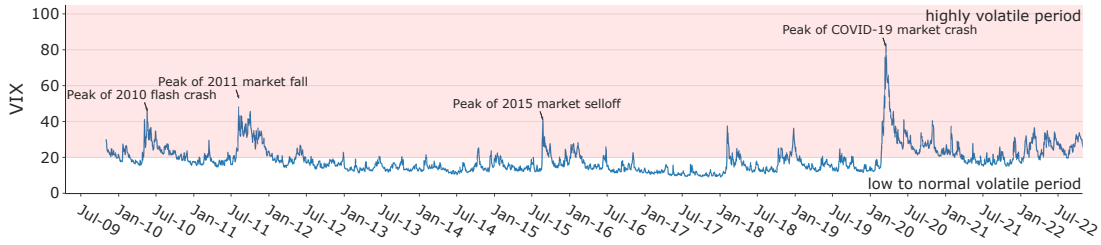
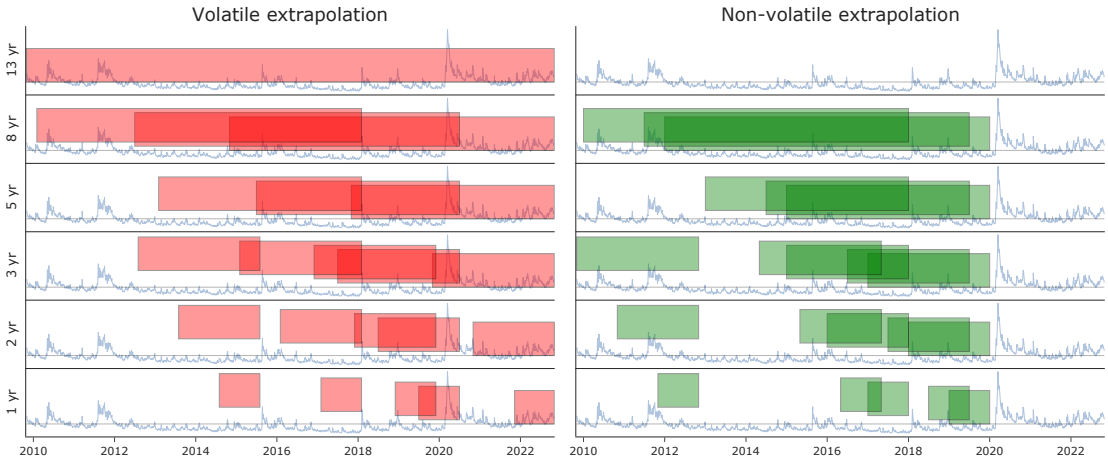


Figure 3.5: Volatility index for the period Nov. 2009 to Nov. 2022 (13 years). The lines that fall in the red region are considered to be highly volatile.

We evaluated data within multiple volatile and non-volatile extrapolation periods, with varying amounts of historical (training) data. Figure 3.6 details the full list of date ranges

[2] investopedia.com/terms/v/vix.asp

employed in our experiments. All but the last five trading days within each period are used as training data. After the training regime, the model is tested on the last five days, which constitute the test set. With unrestricted data access and resources, we could extend our analysis beyond the current 13-year span. This expansion would allow us to cover longer time periods and incorporate more data points, significantly improving the robustness and generalizability of our findings.



(a) Time ranges for volatile and non-volatile extrapolation regime using different amounts of training data. Experiments are done on the same extrapolation period with varying training lengths for comparison.

Training length	Volatile extrapolation	Non-volatile extrapolation
13 years	2009-11:2022-10	-
8 years (x3)	2014-11:2022-10 2012-07:2020-06 2010-02:2018-01	2012-01:2019-12 2011-07:2019-06 2010-01:2017-12
5 years (x3)	2017-11:2022-10 2015-07:2020-06 2013-02:2018-01	2015-01:2019-12 2014-07:2019-06 2013-01:2017-12
3 years (x5)	2019-11:2022-10 2017-07:2020-06 2016-12:2019-11 2015-02:2018-01 2012-08:2015-07	2017-01:2019-12 2016-07:2019-06 2015-01:2017-12 2014-05:2017-04 2009-11:2012-10
2 years (x5)	2020-11:2022-10 2018-07:2020-06 2017-12:2019-11 2016-02:2018-01 2013-08:2015-07	2018-01:2019-12 2017-07:2019-06 2016-01:2017-12 2015-05:2017-04 2010-11:2012-10
1 year (x5)	2021-11:2022-10 2019-07:2020-06 2018-12:2019-11 2017-02:2018-01 2014-08:2015-07	2019-01:2019-12 2018-07:2019-06 2017-01:2017-12 2016-05:2017-04 2011-11:2012-10

(b) 43 different date ranges spanning the market dataset, across volatile and non-volatile market periods.

Figure 3.6: Time ranges used in experiments.

Hyperparameters

All of the training was performed using a compute node with NVIDIA Tesla P100 GPU and 32 GB memory provided by Compute Ontario (Graham) and the Digital Research Alliance of Canada. We previously used preset hyperparameters based on limited preliminary experiments (Olorunnimbe and Viktor, 2022b). In this current work, We present a more extensive hyperparameter search using an open framework called Optuna (Akiba et al., 2019). We initiate the hyperparameter search with the following:

- Learning rate: 0.0001 to 1 in log-step increments
- Dropout rate: 0 to 0.9 in 0.1-step increments
- Current window: 16, 30
- Similarity function: es, kv, cm for SeTT; hd for r-SeTT

Using the hyperparameters above, we ran 200 trials across 43 different timeframes to get optimal hyperparameters for the comparison experiments. Optimal hyperparameters for each of the four models – SeTT(es), SeTT(kv), SeTT(cm), & r-SeTT(hd) – are selected from each of the timeframes, paired with the similarity functions described in Section 3.3.1. Below are listed the best parameters and model combinations across the different timeframes, as determined from the trials. Most of the best parameters were observed within the first 40 trials.

yr length	condition	train_loss	trial_id	dropout	local_win	lr	kind	sim_func
1	non-volatile	0.0637	0	0.3	30	0.082	SeTT	es
1	non-volatile	0.0763	10	0.6	30	0.836	SeTT	es
1	non-volatile	0.0886	35	0.1	30	0.186	SeTT	cm
1	non-volatile	0.0875	1	0.0	16	0.001	SeTT	ks
1	non-volatile	0.0855	18	0.8	30	0.025	SeTT	ks
2	non-volatile	0.0497	17	0.4	30	0.004	SeTT	es
2	non-volatile	0.1054	3	0.9	16	0.016	SeTT	ks
2	non-volatile	0.0655	22	0.2	30	0.183	r-SeTT	hd
2	non-volatile	0.0668	4	0.9	30	0.042	r-SeTT	hd
2	non-volatile	0.0744	4	0.9	30	0.042	r-SeTT	hd
3	non-volatile	0.0691	2	0.2	30	0.001	r-SeTT	hd
3	non-volatile	0.0995	10	0.7	30	0.795	r-SeTT	hd
3	non-volatile	0.0636	14	0.7	30	0.004	SeTT	ks
3	non-volatile	0.0615	0	0.3	30	0.082	SeTT	es
3	non-volatile	0.0812	42	0.0	30	0.620	r-SeTT	hd
5	non-volatile	0.0723	4	0.9	30	0.042	r-SeTT	hd

yr length	condition	train_loss	trial_id	dropout	local_win	lr	kind	sim_func
5	non-volatile	0.1246	3	0.9	16	0.016	SeTT	ks
5	non-volatile	0.0559	2	0.2	30	0.001	r-SeTT	hd
8	non-volatile	0.0693	0	0.3	30	0.082	SeTT	es
8	non-volatile	0.1337	2	0.2	30	0.001	r-SeTT	hd
8	non-volatile	0.0592	18	0.8	30	0.004	SeTT	es
1	volatile	0.0930	6	0.3	30	0.013	SeTT	es
1	volatile	0.0536	4	0.9	30	0.042	r-SeTT	hd
1	volatile	0.0702	26	0.6	30	0.007	r-SeTT	hd
1	volatile	0.0938	37	0.2	30	0.001	SeTT	cm
1	volatile	0.1084	0	0.3	30	0.082	SeTT	es
2	volatile	0.0836	0	0.3	30	0.082	SeTT	es
2	volatile	0.0619	4	0.9	30	0.042	r-SeTT	hd
2	volatile	0.0662	4	0.9	30	0.042	r-SeTT	hd
2	volatile	0.1133	1	0.0	16	0.001	SeTT	ks
2	volatile	0.1022	38	0.2	30	0.001	SeTT	ks
3	volatile	0.0704	0	0.3	30	0.082	SeTT	es
3	volatile	0.0689	4	0.9	30	0.042	r-SeTT	hd
3	volatile	0.0752	4	0.9	30	0.042	r-SeTT	hd
3	volatile	0.0878	2	0.2	30	0.001	r-SeTT	hd
3	volatile	0.1294	45	0.8	30	0.051	r-SeTT	hd
5	volatile	0.0843	32	0.1	30	0.002	r-SeTT	hd
5	volatile	0.0647	4	0.9	30	0.042	r-SeTT	hd
5	volatile	0.0951	3	0.9	16	0.016	SeTT	ks
8	volatile	0.0869	0	0.3	30	0.082	SeTT	es
8	volatile	0.0794	2	0.2	30	0.001	r-SeTT	hd
8	volatile	0.0768	2	0.2	30	0.001	r-SeTT	hd
13	volatile	0.0993	2	0.2	30	0.001	r-SeTT	hd

Table 3.3: Hyperparameters after 200 individual trials across both volatile and non-volatile extrapolation periods and 43 different timeframes.

We compared our model with autoregressive integrated moving average (ARIMA) and generalized autoregressive conditional heteroscedasticity (GARCH), two models commonly used for predictions in finance as a result of their ability to capture the temporal structure and conditional variance in financial time series (R. Hyndman and Athanasopoulos, 2021). One approach to using GARCH for prediction is to fit it on the residual of an ARIMA model, where the GARCH mean-output is the volatility estimate and is added to the ARIMA output (Mustapa and Ismail, 2019). Along with these two financial models, we also compared our models with other deep learning models supporting multi-variate input attributes.

The deep learning models are neural hierarchical interpolation for time series forecast-

ing (NHiTS) (Challu et al., 2023), probabilistic forecasting with autoregressive recurrent networks (DeepAR) (Salinas et al., 2020), RNN (Goodfellow et al., 2016), and our baseline TFT (Lim, Arik, et al., 2021). Both NHiTS and DeepAR are state-of-the-art forecasting models, but they use different techniques to make predictions. NHiTS incorporates hierarchical interpolation and multi-rate data sampling for improved accuracy and faster computation. It was shown to achieve an average accuracy improvement of 20% over basic Transformer architectures. DeepAR trains an autoregressive RNN model on multiple time series data for probabilistic forecasting. It, too, demonstrated improved performance over leading models.

We also compared our model with the TFT model to demonstrate areas of improvement in our approach. As TFT has already been shown to outperform the original Transformer (Lim, Arik, et al., 2021), we excluded it from our current comparison. For completion, we added a sequence-to-sequence RNN model to represent the classical deep learning approach for sequential data. We ran 40 experiments for each model to determine the optimal values for the learning rate and dropout. These hyperparameters can be seen in Table A.1. A total of 15,480 Optuna trials were run to select the optimal hyperparameters for the models, which were determined to be 200×43 for the SeTT algorithms and $40 \times 4 \times 43$ for the other deep learning algorithms.

3.4.2 Results and discussion

In this section, we present the results of our extended work in hyperparameter optimization of the SeTT & r-SeTT algorithms across extended and diversified timeframes. We conducted a series of hyperparameter optimization experiments across various timeframes, as discussed in Section 3.4.1. Our extended work aims to validate previous research and create a foundation for further research based on our approach (Olorunnimbe and Viktor, 2022b). We provide the detailed results below, followed by a discussion of these results.

Results

Table 3.4 presents the weighted-summary results. The MASE was used to evaluate the performance of the SeTT model, in comparison to other state-of-the-art deep learning models and classical financial models, across 43 different volatile and non-volatile periods, using optimized hyperparameters.

The SeTT model always outperformed classical financial models and other state-of-the-art deep learning approaches, further highlighting its efficacy for predicting stock trends.

start	end	yr_len	condition	arima	garch	rnn	deepar	nhits	sett	tft
2011-11-01	2012-11-01	1	non_volatile	1.592 (+105.2%)	1.599 (+106.1%)	0.986 (+27.1%)	0.951 (+22.6%)	1.571 (+102.6%)	**0.776**	0.806 (+4.0%)
2010-11-01	2012-11-01	2	non_volatile	1.54 (+137.8%)	1.545 (+138.7%)	0.963 (+48.8%)	0.799 (+23.4%)	1.353 (+109.0%)	0.656 (+1.3%)	**0.648**
2009-11-01	2012-11-01	3	non_volatile	1.544 (+158.6%)	1.552 (+160.0%)	0.917 (+53.6%)	0.827 (+38.5%)	1.476 (+147.2%)	0.62 (+3.9%)	**0.597**
2016-05-01	2017-05-01	1	non_volatile	1.651 (+134.1%)	1.644 (+133.2%)	0.844 (+19.7%)	**0.705**	0.91 (+29.0%)	0.818 (+16.1%)	0.796 (+12.9%)
2015-05-01	2017-05-01	2	non_volatile	1.567 (+213.1%)	1.556 (+211.0%)	0.701 (+40.1%)	0.647 (+29.3%)	0.864 (+72.5%)	**0.5**	0.55 (+9.8%)
2014-05-01	2017-05-01	3	non_volatile	1.568 (+200.9%)	1.556 (+198.5%)	0.732 (+40.5%)	0.678 (+30.1%)	0.8 (+53.5%)	0.579 (+11.1%)	**0.521**
2017-01-01	2018-01-01	1	non_volatile	1.455 (+89.8%)	1.459 (+90.4%)	0.846 (+10.4%)	**0.767**	0.802 (+4.6%)	0.798 (+4.2%)	0.903 (+17.8%)
2016-01-01	2018-01-01	2	non_volatile	1.361 (+106.9%)	1.365 (+107.6%)	0.752 (+14.4%)	0.701 (+6.6%)	0.672 (+2.2%)	0.675 (+2.7%)	**0.658**
2015-01-01	2018-01-01	3	non_volatile	1.329 (+142.3%)	1.334 (+143.1%)	0.661 (+20.5%)	0.555 (+1.3%)	0.617 (+12.4%)	0.64 (+16.7%)	**0.548**
2013-01-01	2018-01-01	5	non_volatile	1.342 (+149.2%)	1.344 (+149.6%)	0.664 (+23.4%)	0.642 (+19.2%)	0.628 (+16.7%)	0.564 (+4.7%)	**0.538**
2010-01-01	2018-01-01	8	non_volatile	1.31 (+205.0%)	1.311 (+205.2%)	0.601 (+40.0%)	0.593 (+38.2%)	0.592 (+37.9%)	**0.429**	0.511 (+19.0%)
2018-07-01	2019-07-01	1	non_volatile	1.529 (+173.8%)	1.542 (+176.2%)	0.953 (+70.7%)	0.846 (+51.5%)	0.809 (+44.8%)	**0.558**	0.565 (+1.1%)
2017-07-01	2019-07-01	2	non_volatile	1.57 (+130.1%)	1.581 (+131.8%)	0.978 (+43.4%)	0.924 (+35.5%)	0.832 (+21.9%)	**0.682**	0.695 (+1.9%)
2016-07-01	2019-07-01	3	non_volatile	1.638 (+97.8%)	1.641 (+98.2%)	1.145 (+38.3%)	0.963 (+16.4%)	0.868 (+4.8%)	**0.828**	0.829 (+0.1%)
2014-07-01	2019-07-01	5	non_volatile	1.598 (+105.8%)	1.603 (+106.4%)	1.083 (+39.4%)	0.867 (+11.6%)	0.848 (+9.2%)	**0.776**	0.779 (+0.4%)
2011-07-01	2019-07-01	8	non_volatile	1.596 (+141.6%)	1.6 (+142.1%)	1.072 (+62.2%)	0.987 (+49.4%)	0.799 (+20.9%)	**0.661**	0.721 (+9.2%)
2019-01-01	2020-01-01	1	non_volatile	1.302 (+227.9%)	1.305 (+228.6%)	0.525 (+32.1%)	0.46 (+15.8%)	0.672 (+69.1%)	**0.397**	0.399 (+0.5%)
2018-01-01	2020-01-01	2	non_volatile	1.279 (+336.3%)	1.278 (+336.1%)	0.495 (+68.8%)	0.405 (+38.2%)	0.585 (+99.5%)	**0.293**	0.312 (+6.6%)
2017-01-01	2020-01-01	3	non_volatile	1.319 (+219.9%)	1.325 (+221.3%)	0.585 (+42.0%)	0.485 (+17.8%)	0.719 (+74.3%)	0.423 (+2.5%)	**0.412**
2015-01-01	2020-01-01	5	non_volatile	1.31 (+236.5%)	1.308 (+236.1%)	0.508 (+30.4%)	0.49 (+25.9%)	0.681 (+74.9%)	0.419 (+7.6%)	**0.389**
2012-01-01	2020-01-01	8	non_volatile	1.32 (+211.4%)	1.319 (+211.0%)	0.572 (+35.0%)	0.479 (+13.0%)	0.711 (+67.7%)	0.466 (+10.0%)	**0.424**
2014-08-01	2015-08-01	1	volatile	1.737 (+94.8%)	1.737 (+94.8%)	1.207 (+35.4%)	1.048 (+17.5%)	1.392 (+56.0%)	0.896 (+0.5%)	**0.892**
2013-08-01	2015-08-01	2	volatile	1.779 (+93.2%)	1.78 (+93.3%)	1.303 (+41.5%)	1.141 (+23.9%)	1.488 (+61.6%)	**0.921**	1.013 (+10.1%)
2012-08-01	2015-08-01	3	volatile	1.765 (+87.8%)	1.767 (+88.0%)	1.231 (+30.9%)	1.05 (+11.8%)	1.453 (+54.6%)	0.972 (+3.4%)	**0.94**
2017-02-01	2018-02-01	1	volatile	2.253 (+72.8%)	2.256 (+73.0%)	1.502 (+15.2%)	**1.304**	1.644 (+26.1%)	1.347 (+3.3%)	1.362 (+4.5%)
2016-02-01	2018-02-01	2	volatile	2.118 (+85.8%)	2.121 (+86.0%)	1.362 (+19.5%)	1.207 (+5.9%)	1.446 (+26.8%)	**1.14**	1.164 (+2.1%)
2015-02-01	2018-02-01	3	volatile	1.98 (+134.6%)	1.983 (+135.0%)	1.202 (+42.4%)	1.099 (+30.2%)	1.276 (+51.2%)	0.891 (+5.5%)	**0.844**
2013-02-01	2018-02-01	5	volatile	1.981 (+134.8%)	1.984 (+135.2%)	1.253 (+48.5%)	1.062 (+25.9%)	1.224 (+45.0%)	0.883 (+4.7%)	**0.844**
2010-02-01	2018-02-01	8	volatile	1.911 (+153.8%)	1.915 (+154.3%)	1.201 (+59.5%)	1.007 (+33.8%)	1.158 (+53.8%)	**0.753**	0.761 (+1.1%)
2018-12-01	2019-12-01	1	volatile	1.415 (+246.6%)	1.415 (+246.5%)	0.742 (+81.8%)	0.634 (+55.3%)	0.78 (+90.9%)	0.427 (+4.5%)	**0.408**
2017-12-01	2019-12-01	2	volatile	1.413 (+208.0%)	1.41 (+207.3%)	0.763 (+66.3%)	0.652 (+42.1%)	0.742 (+61.6%)	**0.459**	0.471 (+2.6%)
2016-12-01	2019-12-01	3	volatile	1.477 (+182.5%)	1.477 (+182.4%)	0.708 (+35.4%)	0.6 (+14.7%)	0.772 (+47.6%)	**0.523**	0.526 (+0.7%)
2019-07-01	2020-07-01	1	volatile	1.764 (+542.5%)	1.761 (+541.2%)	0.758 (+176.2%)	0.749 (+172.7%)	1.247 (+354.0%)	**0.275**	0.276 (+0.7%)
2018-07-01	2020-07-01	2	volatile	1.917 (+365.9%)	1.915 (+365.4%)	0.972 (+136.3%)	0.907 (+120.5%)	1.561 (+279.3%)	0.417 (+1.4%)	**0.411**
2017-07-01	2020-07-01	3	volatile	2.024 (+318.9%)	2.024 (+318.8%)	1.101 (+127.9%)	0.978 (+102.3%)	1.661 (+243.7%)	**0.483**	0.516 (+6.7%)
2015-07-01	2020-07-01	5	volatile	2.197 (+223.3%)	2.197 (+223.2%)	1.187 (+74.7%)	1.187 (+74.6%)	1.894 (+178.7%)	**0.68**	0.685 (+0.7%)
2012-07-01	2020-07-01	8	volatile	2.317 (+199.3%)	2.317 (+199.3%)	1.345 (+73.7%)	1.314 (+69.8%)	2.069 (+167.3%)	0.815 (+5.3%)	**0.774**
2021-11-10	2022-11-01	1	volatile	1.903 (+277.3%)	1.899 (+276.6%)	0.981 (+94.6%)	0.968 (+91.8%)	1.072 (+112.5%)	**0.504**	0.513 (+1.7%)
2020-11-01	2022-11-01	2	volatile	2.015 (+230.2%)	2.015 (+230.2%)	1.093 (+79.0%)	1.051 (+72.2%)	1.188 (+94.6%)	**0.61**	0.666 (+9.0%)
2019-11-01	2022-11-01	3	volatile	1.872 (+344.7%)	1.866 (+343.2%)	0.964 (+129.0%)	0.876 (+108.2%)	1.074 (+155.2%)	0.436 (+3.6%)	**0.421**
2017-11-01	2022-11-01	5	volatile	1.953 (+261.1%)	1.943 (+259.3%)	1.05 (+94.2%)	1.011 (+86.9%)	1.232 (+127.8%)	0.545 (+0.8%)	**0.541**
2014-11-01	2022-11-01	8	volatile	2.077 (+212.9%)	2.069 (+211.8%)	1.196 (+80.2%)	1.158 (+74.4%)	1.259 (+89.6%)	0.775 (+16.7%)	**0.664**
2009-11-01	2022-11-01	13	volatile	2.131 (+176.9%)	2.124 (+176.0%)	1.264 (+64.3%)	1.2 (+55.9%)	1.295 (+68.2%)	0.782 (+1.7%)	**0.77**

Table 3.4: MASE loss comparison of weighted-summary of our experiments using optimized hyperparameters across 43 timeframes spanning 13 years.

The performance is similar to p50 and p90 comparison metrics. Table 3.5 presents a count of the minimum loss metrics for the different models across the different experiments. The table shows the relative consistency of our model's performance across all timeframes using different evaluation loss functions. Overall, we observed a positive relationship between the error and the length of the historical data window up to 1–3 years, after which it becomes mostly negative.

The key contribution of the SeTT models is that historically similar trends can be incorporated into their design, allowing them to make more accurate predictions in contexts where traditional approaches may be less effective. We conducted extensive optimization

period end	condition	arima	garch	rnn	deepar	nhits	sett	tft
2012-11-01	non_volatile	0	0	0	0	0	1	2
2017-05-01	non_volatile	0	0	0	1	0	1	1
2018-01-01	non_volatile	0	0	0	1	0	1	3
2019-07-01	non_volatile	0	0	0	0	0	5	0
2020-01-01	non_volatile	0	0	0	0	0	2	3
2015-08-01	volatile	0	0	0	0	0	1	2
2018-02-01	volatile	0	0	0	1	0	2	2
2019-12-01	volatile	0	0	0	0	0	2	1
2020-07-01	volatile	0	0	0	0	0	3	2
2022-11-01	volatile	0	0	0	0	0	2	4
	sum	0	0	0	3	0	20	20

(a) MASE loss function

period end	condition	arima	garch	rnn	deepar	nhits	sett	tft
2012-11-01	non_volatile	0	0	0	0	0	1	2
2017-05-01	non_volatile	0	0	0	1	0	1	1
2018-01-01	non_volatile	0	0	0	1	0	1	3
2019-07-01	non_volatile	0	0	0	0	0	3	2
2020-01-01	non_volatile	0	0	0	0	0	2	3
2015-08-01	volatile	0	0	0	0	0	2	1
2018-02-01	volatile	0	0	0	1	0	2	2
2019-12-01	volatile	0	0	0	0	0	2	1
2020-07-01	volatile	0	0	0	0	0	3	2
2022-11-01	volatile	0	0	0	0	0	2	4
	sum	0	0	0	3	0	19	21

(b) p50 loss function

arima	garch	rnn	deepar	nhits	sett	tft
0	0	0	0	0	2	1
0	0	0	0	3	0	0
0	0	0	0	0	5	0
0	0	0	1	0	2	2
0	0	0	2	0	0	3
0	0	0	0	0	2	1
0	0	0	0	0	2	3
0	0	0	0	0	0	3
0	0	0	0	0	5	0
0	0	0	0	0	3	3
0	0	0	3	3	21	16

(c) p90 loss function

Table 3.5: Minimum loss error counts for all models across extrapolation periods and market conditions.

trials for robustness and reliability. The results show that the SeTT model outperformed the baseline TFT model in multiple volatile and non-volatile periods, demonstrating its ability to make more accurate predictions in certain challenging market conditions.

We observed the biggest difference in predictive performance when there were many instances of historical volatility close to the extrapolation period, regardless of the volatility during extrapolation. For example, during the 2019 and 2020 non-volatile extrapolation period, higher instances of performance improvement were observed with the SeTT models compared to the baseline TFT model. For 2019-07, SeTT demonstrated a reduction in error across all historical lengths, as shown in Table 3.5. The SeTT models exhibited the lowest errors for the 2020-01 period, meaning that they provided the best predictions even with a relatively shorter historical market length. This result is indicative of the robustness and veracity of our models and their ability to absorb local trends.

While the weighted performance of SeTT and TFT appear to be equivalent during the periods of volatile extrapolation, a significant difference is observed upon closer examina-

tion of the performance of individual symbols. This raises the important consideration that even if the aggregated performance improvement is marginal, it can be valuable to examine performance improvements at the individual stock level. As demonstrated in Table 3.4, there was a slight overall improvement of less than 1% during the volatile period from July 2019 to July 2020. However, further analysis revealed that some individual stock symbols exhibited significant performance improvements ranging from 5-72%, as presented in Table 3.6. The differences in performance may be attributed to data characteristics specific to each stock, influencing how well the model predicts their movements.

ticker	arima	deepar	garch	nhits	rnn	sett	tft
AAPL	1.617 (+1230.7%)	1.607 (+1222.0%)	0.527 (+333.9%)	0.52 (+327.8%)	0.945 (+678.0%)	**0.122**	0.181 (+49.1%)
AMGN	1.485 (+209.8%)	1.503 (+213.5%)	1.413 (+194.7%)	1.206 (+151.5%)	1.455 (+203.4%)	0.515 (+7.5%)	**0.48**
BA	2.426 (+467.7%)	2.419 (+466.1%)	1.157 (+170.7%)	1.117 (+161.4%)	1.589 (+271.8%)	0.435 (+1.9%)	**0.427**
CAT	1.763 (+612.4%)	1.763 (+612.7%)	0.82 (+231.5%)	0.817 (+230.0%)	1.423 (+475.0%)	0.29 (+17.0%)	**0.247**
CRM	1.683 (+525.8%)	1.679 (+524.3%)	0.971 (+261.2%)	0.806 (+199.8%)	1.075 (+299.9%)	0.315 (+17.2%)	**0.269**
CSCO	1.455 (+696.1%)	1.452 (+694.7%)	0.416 (+127.7%)	0.475 (+159.7%)	0.82 (+348.5%)	**0.183**	0.204 (+11.9%)
CVX	1.832 (+800.3%)	1.834 (+801.2%)	0.446 (+119.4%)	0.485 (+138.3%)	1.21 (+494.8%)	**0.203**	0.249 (+22.5%)
IBM	1.801 (+614.8%)	1.798 (+613.8%)	0.645 (+156.2%)	0.69 (+173.8%)	1.26 (+400.0%)	**0.252**	0.303 (+20.4%)
INTC	1.558 (+1388.3%)	1.568 (+1398.2%)	0.394 (+276.9%)	0.391 (+274.0%)	0.417 (+298.6%)	**0.105**	0.118 (+12.5%)
JNJ	1.629 (+1173.2%)	1.633 (+1176.0%)	0.696 (+444.0%)	0.708 (+452.8%)	1.198 (+836.1%)	**0.128**	0.221 (+72.6%)
KO	1.868 (+571.5%)	1.853 (+566.2%)	0.674 (+142.3%)	0.83 (+198.3%)	1.386 (+398.3%)	0.319 (+14.6%)	**0.278**
MCD	1.632 (+984.9%)	1.635 (+987.3%)	0.351 (+133.4%)	0.467 (+210.7%)	0.78 (+418.4%)	0.225 (+49.6%)	**0.15**
MMM	1.751 (+574.0%)	1.755 (+575.5%)	0.809 (+211.4%)	0.706 (+172.0%)	1.825 (+602.7%)	0.263 (+1.3%)	**0.26**
MSFT	1.66 (+797.8%)	1.654 (+795.0%)	0.575 (+211.3%)	0.568 (+207.5%)	1.142 (+517.8%)	**0.185**	0.194 (+4.9%)
NKE	2.362 (+739.8%)	2.347 (+734.6%)	0.908 (+222.9%)	0.968 (+244.2%)	1.306 (+364.4%)	0.336 (+19.3%)	**0.281**
PG	1.857 (+471.7%)	1.853 (+470.3%)	0.866 (+166.5%)	0.913 (+181.1%)	1.151 (+254.3%)	0.36 (+10.8%)	**0.325**
UNH	1.669 (+554.9%)	1.656 (+549.9%)	0.695 (+172.9%)	0.749 (+194.0%)	1.311 (+414.4%)	**0.255**	0.327 (+28.2%)
V	1.672 (+718.3%)	1.658 (+711.6%)	0.572 (+179.9%)	0.52 (+154.4%)	1.481 (+625.0%)	**0.204**	0.221 (+8.1%)
VZ	2.103 (+495.3%)	2.098 (+494.1%)	0.994 (+181.4%)	1.04 (+194.4%)	1.443 (+308.5%)	0.395 (+11.9%)	**0.353**
WMT	1.449 (+550.4%)	1.45 (+551.0%)	0.524 (+135.4%)	0.579 (+159.7%)	0.541 (+143.0%)	0.234 (+5.1%)	**0.223**

Table 3.6: MASE loss comparison of individual stock symbols for the volatile period 2019-07 to 2020-07

As previously discussed, the impact of long-distance volatility on predictions appears to be less pronounced when extrapolating during volatile periods. Although the exact cause of this discrepancy remains uncertain, it might be attributed to a preference for a shorter historical timeframe caused by the inherent volatility of the period.

To evaluate the difference in error distribution between the various models, we conducted a Mann-Whitney-Wilcoxon test (Paquet and Soleymani, 2022). The Mann-Whitney-Wilcoxon test, a non-parametric statistical method, allows for the comparison of error distribution across different models without making assumptions about the underlying distributions. The results of this test allowed us to reject the null hypothesis for the classical financial models and the state-of-the-art deep learning methods, indicating that our models have significant predictive performance improvement compared to the traditional models. For comparison between SeTT and TFT models, the periods in which the null hypothesis

is rejected correspond to the periods with a significant loss improvement in Table 3.4.

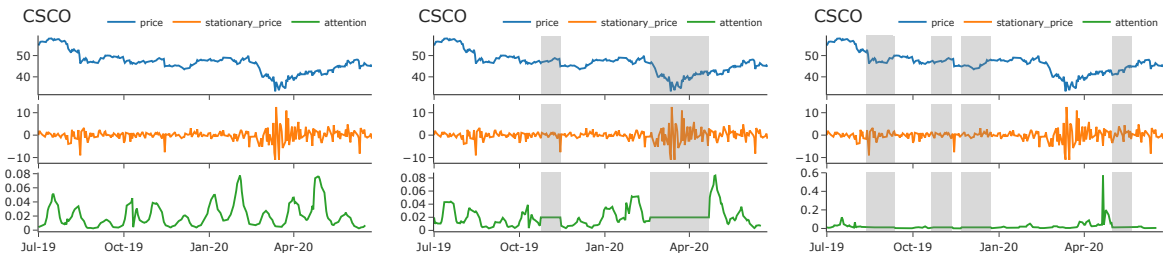
	mase	p50	p90
SeTT/ARIMA	0.000002	0.000002	0.000002
SeTT/GARCH	0.000002	0.000002	0.000002
SeTT/RNN	0.000002	0.000002	0.000004
SeTT/DeepAR	0.000004	0.000004	0.000105
SeTT/NHiTS	0.000004	0.000004	0.010689
SeTT/TFT	0.023951	0.026642	0.032768

Table 3.7: Comparison of error distribution using a non-parametric Mann–Whitney test for the volatile period 2020-10 to 2020-10.

An example of such comparison can be seen in Table 3.7 for a highly volatile period between Nov. 2020 and Nov. 2022 with a weighted performance improvement of 9%. Overall, the results of this test provide strong evidence for the improved performance of our models compared to the baseline models.

Discussion

The results presented in Section 3.4.2 demonstrate the robustness and generalisability of our core research idea, as our model performs consistently across a wide range of timeframes and market conditions. Although the hyperparameters of the comparison models were optimized to achieve the best results under the test conditions, our model outperformed the others during volatile periods and maintained competitiveness in non-volatile periods.



(a) Unmuted attention trend (TFT)

(b) Attention trend with dissimilar regions (SeTT)

(c) Attention trend with dissimilar regions (r-SeTT)

Figure 3.7: Attention on the historical window for CSCO (Cisco Systems, Inc.) - 2019-07 to 2020-07.

Using the period and result details in Table 3.6 as an illustrative example, it is evident that without the similarity vector, our model runs the risk of focusing on an incorrect timeframe during the sample extrapolation regime. As Figure 3.7a shows, the cost of that misplaced focus is a loss of predictive performance. However, Figures 3.7b and 3.7c demonstrate that with the similarity vector, the model focuses on the timeframes that are most similar to the current window and the model makes better extrapolations as a result.

Overall, the findings of this study provide strong evidence for the effectiveness of our approach in predicting stock trends and demonstrate its potential value as a useful tool in this domain. Further research may be needed to explore the full capabilities of the SeTT architectures and to identify any potential limitations or areas for improvement. However, the results of this study offer a promising foundation for continued research and practical applications in financial time series forecasting.

3.5 Summary

In this chapter, we introduced novel temporal Transformer architectures based on sliding windows in the context of financial time series, namely SeTT. We demonstrated the extrapolation performance of SeTT models through a comprehensive and rigorous evaluation process focused on financial forecasts. We conducted over 15,000 optimization trials across 13 years and 43 different timeframes and used the optimized hyperparameters to compare the SeTT models to other state-of-the-art models for time series data. Using static metadata for 20 symbols from the DJIA index allowed us to evaluate model performance in a range of market conditions and industries. Augmented by the similarity vector, the SeTT model consistently outperforms the others in a range of market conditions and provides improvements over the performance of the TFT in most conditions. The ability to incorporate historically similar trends into the temporal Transformer's architecture contributes to its improved performance with multi-horizon time series forecasts, when traditional approaches may be less effective.

While the TFT model also performs well and competes favorably with the SeTT models, it is unclear which model is superior across various market conditions. This highlights the importance of ongoing research and development in this regard. Rather than choosing a single "winning" model, in the next chapter, we will explore the approach of combining both model architectures in a deep-learning ensemble. This approach would take advantage of the strengths of both models and potentially achieve improved performance for multi-horizon financial time series forecasting. Overall, the extensive optimization trials

conducted in this study demonstrated the robustness and reliability of our model, establishing its value as a financial forecasting tool. The results of this study offer a promising foundation for the continued development and use of the similarity vector, and indeed the SeTT model in the field of financial analysis.

In the next chapter, we introduce an ensemble-based method that extends the SeTT and r-SeTT architectures.

Chapter 4

Efficient utilization of extended time series via ensemble models

This chapter represents our research paper, namely:

- *Ensemble of temporal Transformers for financial time series* accepted for publication in the Journal of Intelligent Information Systems (Olorunnimbe and Viktor, 2024) (Olorunnimbe and Viktor, 2024b).
- In this paper, we present the Windowing ensemble of temporal Transformers (WETT) approach that efficiently utilizes the available data over an extended timeframe. Our ensemble combines multiple temporal Transformer models learned within sliding windows, thereby making optimal use of the data. We observed significant improvements in predictive performance compared to the baseline temporal Transformer models.

4.1 Overview

In Chapter 2, we itemized research questions related to deep learning in the financial domain and summarized some works of research answering those questions. We prioritized works that have been backtested, suggesting that some due diligence has been done regarding the practicality and reproducibility of the research work. While acknowledging that more attention has been given to this area of study in recent years, it is clear that numerous research challenges remain. Based on the findings, we introduce temporal Transformer-based architectures for financial forecast in Chapter 3. These architectures incorporate this

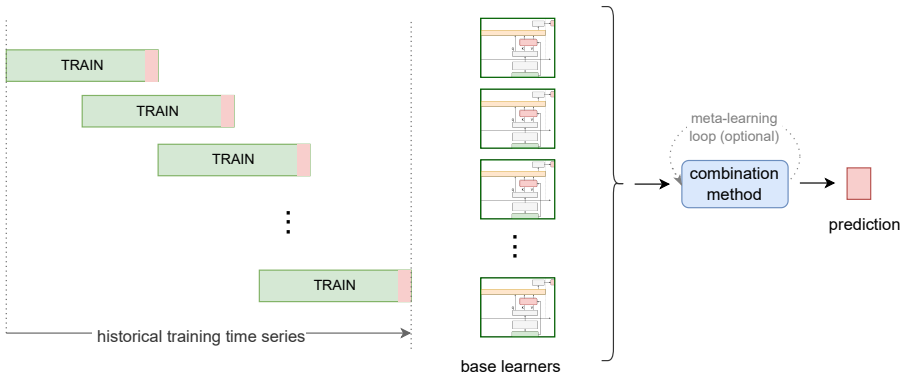
characteristic into the Transformer's attention mechanism, and we demonstrated their improved performance across numerous timeframes and intervals. However, the performance varies across different time scenarios. Additionally, recall that in Chapter 3, we found that despite having a sufficiently extensive time series, all temporal Transformers employed in our study fail to fully utilize the entirety of the time series data. Rather most of the time, the optimal historical timeframe ranging from 1 to 3 years of historical data.

In this chapter, we present a novel approach that utilizes ensembles of these shorter timeframes to create base models for use in an ensemble of temporal Transformer models. Ensemble learning employs several individual models commonly called *base models*. It combines their predictions by averaging, voting, or using another machine learning process to learn the optimal way of combining these base models. This is primarily done to reduce the bias and variance in the individual base learners (Russell and Norvig, 2021), thereby achieving a bias-variance tradeoff that improves the performance of the overall ensemble model allowing for better generalization to unseen data. This is advantageous for complex processes such as deep learning, where individual models are susceptible to overfitting the training data (Ganaie et al., 2022; Russell and Norvig, 2021). In ensemble learning, a common approach for combining the predictions of the base models for regression problems such as this is to take the average of the individual models' outputs (Russell and Norvig, 2021). We also explored a meta-learning or stacking approach to learn the optimal weights for combining the outputs of the base models.

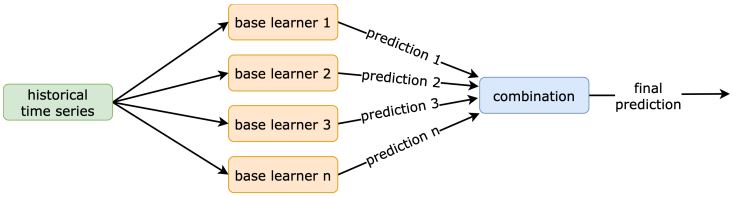
4.2 Ensemble of temporal Transformers

We present our approach that combines SeTT, r-SeTT, and TFT, in ensemble models, with a focus on improved performance for financial data. Figure **??** illustrates how the ensemble model is constructed from sliding windows of financial time series data. We pre-trained the base models and used a model registry to persist the models from the windowing history for consistency and model reusability and to simplify model development lifecycle (A. Chen et al., 2020). Given the considerable training time required by Transformer models, our approach of creating a windowing ensemble not only improves the predictive performance but also enhances computational efficiency without sacrificing the accuracy of the model, as will be demonstrated in our results. Using the same model of a financial time series from Section 3.3.1, $X_{t,t\in\{1,\ldots,T,T+1,\ldots,T+\tau\}}$, and targets $y_{t,t\in\{1,\ldots,T,T+1,\ldots,T+\tau\}}$ serves as extended historical time series features and targets used as input to our ensemble approach, named WETT.

The last τ days of the overall historical time series are excluded, and the rest is used



(a) Illustration of base learners generated from sliding window of historical data



(b) Simplified representation of ensemble model

Figure 4.1: Ensemble of base models using overlapping time series from the extended historical time series. Stacking is optionally used to learn the weights of the weight learners in a meta-learning loop.

to create the base models of the ensemble. The base models are generated from sliding windows of the remaining historical time series, the combination methods, as well as the diversity of the constituent models. For the ensemble model, the τ days unseen by the base learners are used as a test set. Algorithm 4.1 depicts the pseudocode of the WETT algorithm.

4.2.1 Windowing historical timeframes

Temporal data is inherently sequential, and the order of the data plays a significant role in the performance of predictive models. Financial market data is no exception, as it is characterized by high volatility and non-constant variance over time, making it challenging to develop models that can make accurate predictions. We incorporate the temporal information into predictive models using sliding windows of historical time series as input to base learners of an ensemble model. The input to each base learner is a fixed-length

Algorithm 4.1 Windowing ensemble of temporal Transformers (WETT)

Input:

$\mathbf{X} = X_t : t \in 1, 2, ..., T$ ▷ full historical time series features of size T

$\mathbf{y} = y_t : t \in 1, 2, ..., T$ ▷ full historical time series targets (Δ`price`) of size T

s, l ▷ sizes of sliding historical timeframe & current window ($l < \mathsf{s} < T$)

b ▷ base learner (TFT, SeTT, r-SeTT, or all)

d ▷ base diversity (none, model, or seed)

c ▷ combination method (average, q-stack)

Output:

$\hat{y} \sim \{y_\tau \mid \tau \in \{1, \ldots, \tau\}\}$ ▷ vector of τ future quantile forecasts

$M \leftarrow \emptyset, L \leftarrow \emptyset$ ▷ set of base models and base current windows

for $i \leftarrow 1$ **to** $T - \tau$, step $\mathsf{s}/2$ **do**

  $X_i, y_i \leftarrow \{\mathbf{X}, \mathbf{y}\}_{t=i}^{i+\mathsf{s}+1}$ ▷ sliding window features/targets sets

  $L \leftarrow L \cup \{y_{i,(l:)}\}$

  $M_i \leftarrow$ train_base(X_i, y_i) ▷ multiple base models if d is model or seed

  $M \leftarrow M \cup \{M_i\}$

note: model registry is employed for reusability

$X_j, y_j \leftarrow \{\mathbf{X}, \mathbf{y}\}_{t=T-\mathsf{s}+1}^{T}$

if c = average **then**

  $\hat{y} \leftarrow$ AVERAGE(X_j, y_j, M)

else if c = q-stack **then**

  $\hat{y} \leftarrow$ Q_STACK(X_j, y_j, M, τ)

window consisting of a sequence of data points. The ensemble model is then used to predict a future horizon unseen by the base models.

Several studies have shown the effectiveness of using a sliding window of financial market time series data as input to deep learning algorithms. For instance, M. Wen et al. uses a sliding window to capture latent patterns from financial time series as CNN input to predict stock market trends, outperforming other deep learning and traditional statistical models (M. Wen et al., 2019). Similarly, we demonstrated that temporal Transformer models trained on a relatively short period of 1-3 years of financial market time series data achieved better performance than models trained on more than three years in Chapter 3.

We posit that by combining an ensemble of a shorter period over an extended historical timeframe, we can more efficiently utilize the full data across the different market condi-

tions, and by implication improve the overall model performance. The ensemble models use an extended historical time frame of 10 to 13 years. A research work proposes utilizing a maximum of $2,739$ data points (trading days), equivalent to 10.87 years, as input for a financial trading model, with a recommendation of 681 data points (2.71) for optimal results (Chan, 2021). Also, Lopez de Prado suggests that researchers should have at least two years of historical data to achieve the best possible outcome (Lopez de Prado, 2013; Prado, 2018). Motivated by these studies, and by our finding of a 1-3 optimal period for a temporal Transformer in Chapter 3, multiple 2-year windowing timeframes, $X_t, y_{t,t\in\{1,\ldots,\mathtt{s}\}}$, in strides of 1-year, are created from the extended historical features $X_t, y_{t,t\in\{1,\ldots,T\}}$. These sliding timeframes are used to create independent base models, each with its current window, $y_l \leftarrow y_{t,t\in\{T-l+1,\ldots,\mathtt{s}\}}$, and forecast horizon, $\hat{y} \sim \{y_\tau \mid \tau \in \{1,\ldots,\tau\}\}$. These base models are used to construct the ensemble model.

4.2.2 Ensemble diversity

In ensemble learning, diversity refers to the extent of dissimilarity among the base learners in terms of training data, algorithms, hyperparameters, or architectures (Ganaie et al., 2022). Common strategies for diversifying ensemble models include bootstrap aggregation, or *bagging*, and boosting (Russell and Norvig, 2021). Bagging involves generating distinct training samples for the base models by performing bootstrap resampling with replacement from the original training set. Boosting, on the other hand, trains models sequentially, with each new model focusing on the errors made by previous ones, based on weight adaptation, to enhance overall accuracy. We diversify our WETT model by employing different sliding windows of the timeframe in our base models. This approach ensures that each base model receives a distinct subset of the original training set as input, thereby introducing diversity in a manner that is comparable to the bagging technique.

We also explore the option of diversifying the base models using different numbers of seeds to initialize the weights to the same base model. According to Goodfellow et al., random initialization differences often cause models trained on the same data set to make partially independent errors (Goodfellow et al., 2016). It has also been shown that non-bootstrap ensembles perform well in practice, sometimes better than the bootstrap approach (Fort et al., 2020). Fort et al. demonstrated that the local minima from different randomly initialized weights of a neural network model are very diverse with unique decision boundaries (Fort et al., 2020). Diversifying the base models by initialization weights can help mitigate the risk of getting stuck in a local minimum and improve the robustness of the ensemble. For our weight-diversified ensemble models, we employed three different initialization seeds corresponding to the number used by Fort et al. (Fort et al., 2020).

However, diversifying by initialization weight may only sometimes be effective since the base models are still based on the same architecture and may not address the limitations specific to the architecture. To this end, we also considered another method for model diversification using different types of base models. The intuition behind diversifying the base models is that different models capture various aspects of the underlying data, and thus their combination is expected to lead to better performance. Using too many types of models, however, can lead to a decrease in performance due to the difficulty in combining diverse outputs.

4.2.3 Combination methods

We employ two methods for combining the prediction of the base models generated from the sliding window of historical timeframes, namely average and q-stack combinations.

Average combination

The average combination method is a baseline approach in ensemble learning (Russell and Norvig, 2021). Multiple neural networks achieve a variety of solution points, such that they can also benefit from model averaging when trained on the same data set (Goodfellow et al., 2016). The ensemble prediction is computed as the average of all the base models' predictions:

$$\hat{y}_{ens} = \frac{1}{|M|} \sum_{i=1}^{|M|} M_i(X, y) \tag{4.1}$$

This combination method has been shown to perform well in many ensemble models, including bagging and random forest, a variant of bagging that combines multiple decision trees on the bootstrap samples to improve performance (Ganaie et al., 2022).

Q-stack combination

Stacked generalization (stacking) is another way of creating ensembles by training a secondary model, called a meta-learner, to make a final prediction based on the aggregated outputs from the base models (Russell and Norvig, 2021). In Section 3.3.1, we discussed

quantile loss as a robust loss function for multi-horizon forecast over a probabilistic range. Building on that, we introduce *q-stack* meta-learner which is a combination method that uses quantile regression neural network (QRNN) (Taylor, 2000), a feedforward neural network used for quantile forecast. We use QRNN to learn the optimal set of weights $\mathbf{w} = \{w_1, w_2, \cdots\}$ by minimizing the quantile loss of the meta-learner across the validation forecast horizon.

$$\mathbf{w} = argmin_{\mathbf{w}} \sum_{i} \rho_{\tau}(y_i - \sum_{j} w_j \hat{y}_{i,j}) \tag{4.2}$$

where $\hat{y}_{i,j}$ is the prediction of the j-th base learner for the i-th instance, $\rho\tau$ is the weighting function for the quantile loss. The computed weights are combined with the base learners for the final prediction.

$$\hat{y}_{ens} = \sum_{i=1}^{|M|} w_i M_i(X, y) \tag{4.3}$$

The stacking meta-learning method has been shown to perform well in many ensemble models (Ganaie et al., 2022). The q-stack specifically is motivated by QRNN's robustness to outliers, and its ability to estimate nonlinear models at different quantiles (Pradeepkumar and Ravi, 2017; Taylor, 2000).

Algorithm 4.2 depicts the combination options used in our ensemble models.

Algorithm 4.2 WETT combination methods

Input:

$\mathbf{X} = X_t : t \in 1, 2, ..., \mathtt{s}$ ▷ sliding window time series features of size $\mathtt{s}$
$\mathbf{y} = y_t : t \in 1, 2, ..., \mathtt{s}$ ▷ sliding window time series targets ($\Delta\mathtt{price}$) of size $\mathtt{s}$
l ▷ size of current window ($l < \mathtt{s}$)
M ▷ set of base models
L ▷ set of l targets used with each base model
τ ▷ size of forecast horizon
k ▷ number of similar base models (optional)

procedure AVERAGE($\mathbf{X}, \mathbf{y}, M$)
    return $\frac{1}{|M|} \sum_{m \in M} m.\text{predict}(\mathbf{X}, \mathbf{y})$

procedure Q_STACK($\mathbf{X}, \mathbf{y}, M, \tau$)
    $\mathbf{w} \leftarrow \left\{ \frac{1}{|M|}, \frac{1}{|M|}, \ldots, \frac{1}{|M|} \right\}$ ▷ initialize equal weight for each base model
    $X_i, y_i \leftarrow \{\mathbf{X}, \mathbf{y}\}_{t=1}^{\mathtt{s}-\tau}$
    repeat
        $\mathbf{y}' \leftarrow \forall_{m \in M}\ m.\text{predict}(X_i, y_i)$
        $\boldsymbol{\Delta} \leftarrow y_i - (\mathbf{y}' \cdot \mathbf{w})$
        $\mathbf{w} \leftarrow \mathbf{w} + \rho_\tau \cdot \boldsymbol{\Delta} \cdot \mathbf{y}'$ ▷ ρ_τ is the quantile loss across τ
    until stopping condition
    $\mathbf{y}'' \leftarrow \forall_{m \in M}\ m.\text{predict}(\mathbf{X}, \mathbf{y})$
    return $\sum(\mathbf{y}'' \cdot \mathbf{w})$

For each windowing timeframe of training data, we created ensemble models using single architecture base models of either TFT, SeTT or r-SeTT, initialized with three seeds, as motivated by Fort et al. For comparison, we also created an ensemble of all three TFT, SeTT, and r-SeTT architectures as individual base models, each initialized with one seed. Below in Table 4.1 are the different variants of the ensemble based on diversification and choice of combination method.

We also created single-seed initialized variants of the single-architecture variants, i.e. WETT.a.tft.sw, for use in an ablation study.

	multi-seed weight initialized	single-seed weight initialized
average combination	WETT.a.tft.mw WETT.a.sett.mw WETT.a.rsett.mw	WETT.a.tft.sw WETT.a.sett.sw WETT.a.rsett.sw WETT.a.all.sw
q-stack combination	WETT.q.tft.mw WETT.q.sett.mw WETT.q.rsett.mw	WETT.q.tft.sw WETT.q.sett.sw WETT.q.rsett.sw WETT.q.all.sw

Table 4.1: Matrix table showing the names of the different WETT variants along with their combination methods and initialization weight options.

4.3 Experimental evaluation

We constructed data sets consisting of information about the 20 companies shown in Table 3.1 from Chapter 3 across multiple volatile and non-volatile extrapolation periods.

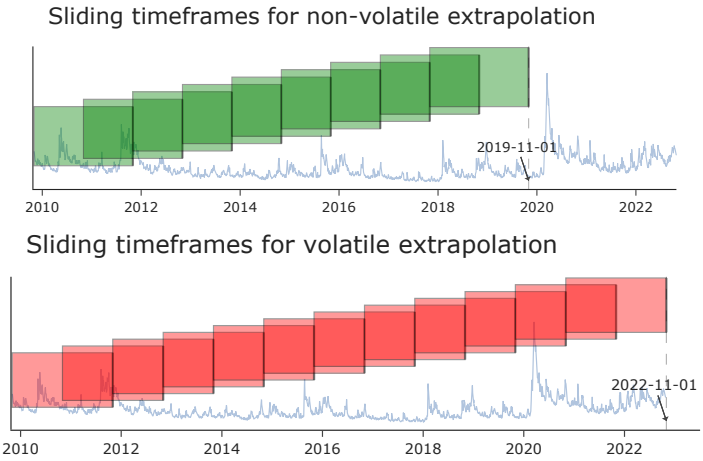


Figure 4.2: Sliding window time ranges used in the ensemble base models for both extrapolation periods. The windows are constructed from period Nov 2009 to Nov 2022 (13 years).

Figure 4.2 shows the windowing timeframes used across both periods. In Section 4.2.1 we discussed using a window length of 2 years, in strides of 1 year. All but the forecast

horizon are used as historical training data within each period. With historical lengths of 10 and 13 years across the non-volatile and volatile market condition based on the available data, a total of 9 and 12 windows were derived, respectively. Employing 3 distinct seeds for the multi-seed initialized variants results in 27 and 36 base learners for the respective market condition. Likewise, we have the same number of base learners for the single-seed initialized variable using all 3 temporal Transformer algorithms. In the case of the single-seed initialized variable, employing all 3 temporal Transformer algorithms results in equivalent numbers of base learners. After training the base learners, the ensemble model is tested on the forecast horizon that constitutes the test set.

4.3.1 Experimentation setup

Recall that in Section 3.4.1, we conducted an extensive hyperparameter search. Based on the findings of this search, we used the following hyperparameters for the rest of the experiment: learning rate of 0.018, dropout rate of 0.3, and current window of 30 days. Also, we used *es* and *hd* for the similarity function of the base learner SeTT and r-SeTT base learners respectively. The input into the base models consists of independent 2-year windowing timeframes, split into non-overlapping training and test sets. The usage of 2 years is motivated by the findings that 1-3 years of financial market time series data achieved better performance than models trained on more than 3 years (Chapter 3). For each 2-year input timeframe, the last 5 days are used as a test set, while the rest is used for training, including a 30-day current window, l.

For baseline comparisons, we compared with ARIMA and GARCH, two models commonly for prediction in finance as a result of their ability to capture the temporal structure, as well as the conditional variance in financial time series (R. Hyndman and Athanasopoulos, 2021). One approach when using GARCH for prediction is to fit it on the residual of an ARIMA model, where the GARCH mean-output is the volatility estimate, and it is added to the ARIMA output (Mustapa and Ismail, 2019). Along with these 2 financial models, we also compared our models with other deep learning models supporting multi-variate input attributes.

The deep learning architectures used in our comparison are NHiTS (Challu et al., 2023), DeepAR (Salinas et al., 2020), RNN (Goodfellow et al., 2016) and our baseline TFT (Lim, Arik, et al., 2021). NHiTS and DeepAR are state-of-the-art forecasting models that use different techniques to make predictions. NHiTS incorporates hierarchical interpolation and multi-rate data sampling for improved accuracy and faster computation, and it was shown to achieve an average accuracy improvement of 20% over basic Transformer architectures.

DeepAR trains an autoregressive RNN model on multiple time series data for probabilistic forecasting, and it also demonstrated improved performance over leading models.

4.3.2 Results and discussion

In this section, we present the results of our experiments comparing the different variants of WETT and the baseline comparison models, across historical time series, with volatile and non-volatile extrapolation periods. The comparison loss metric pertains to unseen test data, rather than training loss.

Baseline comparison

We compared our models to other financial and deep learning models. As shown in Table 4.2, our ensemble models consistently outperform all the baseline models. The percentage values represent the performance degradation of the respective model when compared with the optimal ensemble model for the same evaluation period. Our ensemble models consistently outperform other comparison models and also offer significant improvements over non-ensemble versions of the TFT model - we observed a performance improvement of more than 60% in a highly volatile extrapolation period. Specifically, when using the WETT-TFT ensemble variant with the average-combination method, there is a substantial performance improvement of 62.2%, when compared with using the standalone TFT model.

Note that in certain cases, the training duration, measured in seconds, is considerably higher for ensembles, especially when using a meta-learner, compared to baseline models. However, the improvement in predictive performance justifies this trade-off in this experimentation setup. Furthermore, as computational resources become more affordable and accessible, solutions that employ neural processing Units (NPUs) could be used to reduce the training time. Nevertheless, augmenting computational resources will not rectify the performance deficit for poor-performing models, without fundamentally changing the underlying algorithms.

Although WETT-TFT demonstrate strong performance among the aggregated results of the ensemble models as shown in Table 4.2, individual stock analysis suggests that the WETT-SeTT & WETT-r-SeTT variants are preferred. These variants prove to be more optimal based on the granular results, as they consistently outperform the WETT-TFT model in greater evaluation instances, as shown in Tables 4.3 & 4.4.

start	end	yrs	condition	combination	model	duration (sec)	loss
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.all.sw	953.31	0.496 (-17.2%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.rsett.mw	880.59	0.466 (-11.9%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.sett.mw	1120.14	0.429 (-4.3%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.tft.mw	859.07	**0.411**
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.all.sw	117.05	0.765 (-46.3%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.rsett.mw	43.0	0.449 (-8.4%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.sett.mw	283.35	0.419 (-1.9%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.tft.mw	26.27	0.486 (-15.5%)
2009-11-01	2019-11-08	10	non-volatile	-	arima	93.6	0.683 (-39.9%)
2009-11-01	2019-11-08	10	non-volatile	-	deepar	41.34	0.793 (-48.2%)
2009-11-01	2019-11-08	10	non-volatile	-	garch	92.94	0.679 (-39.5%)
2009-11-01	2019-11-08	10	non-volatile	-	nhits	16.11	15.326 (-97.3%)
2009-11-01	2019-11-08	10	non-volatile	-	rnn	41.31	0.767 (-46.5%)
2009-11-01	2019-11-08	10	non-volatile	-	tft	193.2	0.701 (-41.4%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.all.sw	1221.96	0.556 (-7.1%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.rsett.mw	1140.77	0.616 (-16.1%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.sett.mw	1455.4	0.618 (-16.4%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.tft.mw	1107.62	1.331 (-61.2%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.all.sw	155.5	0.582 (-11.2%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.rsett.mw	55.92	0.545 (-5.1%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.sett.mw	380.14	0.543 (-4.9%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.tft.mw	34.13	**0.517**
2009-11-01	2022-11-08	13	volatile	-	arima	205.19	0.987 (-47.7%)
2009-11-01	2022-11-08	13	volatile	-	deepar	55.52	1.17 (-55.8%)
2009-11-01	2022-11-08	13	volatile	-	garch	205.77	0.986 (-47.6%)
2009-11-01	2022-11-08	13	volatile	-	nhits	21.34	55.835 (-99.1%)
2009-11-01	2022-11-08	13	volatile	-	rnn	55.44	1.226 (-57.9%)
2009-11-01	2022-11-08	13	volatile	-	tft	202.12	1.368 (-62.2%)

Table 4.2: Comparison of WETT models with baseline financial and deep learning models, demonstrating that WETT consistently outperforms the baseline models.

In Table 4.3, the individual stocks were evaluated during the non-volatile extrapolation period from Nov 2009 to Nov 2019. Within this timeframe, there are 13 instances where the WETT-SeTT & WETT-r-SeTT variants are the optimal models. The WETT-TFT variants are optimal 7 times. A similar observation can be seen in Table 4.4 during the volatile extrapolation period.

ticker	WETT.a.all.sw	WETT.a.rsett.mw	WETT.a.sett.mw	WETT.a.tft.mw	WETT.q.all.sw	WETT.q.rsett.mw	WETT.q.sett.mw	WETT.q.tft.mw	arima	deepar	garch	nhits	rnn	tft
AAPL	0.65 (-76.9%)	0.19 (-21.1%)	0.17 (-11.8%)	0.33 (-54.5%)	0.27 (-44.4%)	0.27 (-44.4%)	0.17 (-11.8%)	**0.15**	0.55 (-72.7%)	0.64 (-76.6%)	0.54 (-72.2%)	30.68 (-99.5%)	0.64 (-76.6%)	0.56 (-73.2%)
AMGN	0.75 (-44.0%)	0.46 (-8.7%)	0.48 (-12.5%)	0.45 (-6.7%)	0.65 (-35.4%)	0.6 (-30.0%)	0.6 (-30.0%)	**0.42**	0.69 (-39.1%)	0.85 (-50.6%)	0.68 (-38.2%)	24.69 (-98.3%)	0.64 (-34.4%)	0.7 (-40.0%)
BA	0.74 (-23.0%)	**0.57**	0.6 (-5.0%)	0.58 (-1.7%)	0.68 (-16.2%)	0.63 (-9.5%)	0.71 (-19.7%)	0.63 (-9.5%)	0.88 (-35.2%)	1.06 (-46.2%)	0.87 (-34.5%)	10.11 (-94.4%)	1.13 (-49.6%)	0.85 (-32.9%)
CAT	0.72 (-70.8%)	**0.21**	0.28 (-25.0%)	0.34 (-38.2%)	0.32 (-34.4%)	0.26 (-19.2%)	0.22 (-4.5%)	**0.21**	0.95 (-77.9%)	0.89 (-76.4%)	0.94 (-77.7%)	16.31 (-98.7%)	0.73 (-71.2%)	0.6 (-65.0%)
CRM	0.46 (-58.7%)	0.27 (-29.6%)	0.27 (-29.6%)	0.33 (-42.4%)	0.23 (-17.4%)	0.26 (-26.9%)	**0.19**	0.22 (-13.6%)	0.39 (-51.3%)	0.74 (-74.3%)	0.38 (-50.0%)	15.78 (-98.8%)	0.57 (-66.7%)	0.52 (-63.5%)
CSCO	0.87 (-69.0%)	**0.27**	0.37 (-27.0%)	0.58 (-53.4%)	0.52 (-48.1%)	0.3 (-10.0%)	0.38 (-28.9%)	0.46 (-41.3%)	0.51 (-47.1%)	0.69 (-60.9%)	0.5 (-46.0%)	126.46 (-99.8%)	0.9 (-70.0%)	0.62 (-56.5%)
CVX	1.09 (-41.3%)	**0.64**	0.72 (-11.1%)	0.75 (-14.7%)	0.93 (-31.2%)	0.73 (-12.3%)	0.82 (-22.0%)	0.82 (-22.0%)	1.19 (-46.2%)	1.28 (-50.0%)	1.17 (-45.3%)	35.78 (-98.2%)	1.44 (-55.6%)	0.92 (-30.4%)
IBM	0.99 (-58.6%)	0.53 (-22.6%)	0.65 (-36.9%)	0.75 (-45.3%)	0.46 (-10.9%)	0.53 (-22.6%)	**0.41**	0.52 (-21.2%)	0.71 (-42.3%)	0.44 (-6.8%)	0.7 (-41.4%)	1.22 (-66.4%)	0.44 (-6.8%)	0.68 (-39.7%)
INTC	0.74 (-74.3%)	0.26 (-26.9%)	**0.19**	0.46 (-58.7%)	0.34 (-44.1%)	0.28 (-32.1%)	0.2 (-5.0%)	0.22 (-13.6%)	0.41 (-53.7%)	0.45 (-57.8%)	0.41 (-53.7%)	29.58 (-99.4%)	0.69 (-72.5%)	0.59 (-67.8%)
JNJ	0.97 (-56.7%)	0.45 (-6.7%)	0.47 (-10.6%)	0.49 (-14.3%)	0.67 (-37.3%)	0.71 (-40.8%)	0.58 (-27.6%)	**0.42**	0.46 (-8.7%)	0.85 (-50.6%)	0.46 (-8.7%)	5.2 (-91.9%)	0.82 (-48.8%)	0.83 (-49.4%)
KO	0.7 (-50.0%)	0.86 (-59.3%)	0.48 (-27.1%)	0.62 (-43.5%)	**0.35**	0.51 (-31.4%)	0.4 (-12.5%)	0.51 (-31.4%)	1.11 (-68.5%)	0.81 (-56.8%)	1.12 (-68.8%)	33.14 (-98.9%)	0.69 (-49.3%)	0.57 (-38.6%)
MCD	1.34 (-43.3%)	0.92 (-17.4%)	0.87 (-12.6%)	0.88 (-13.6%)	1.05 (-27.6%)	0.86 (-11.6%)	0.84 (-9.5%)	**0.76**	1.48 (-48.6%)	1.11 (-31.5%)	1.48 (-48.6%)	28.95 (-97.4%)	1.19 (-36.1%)	1.13 (-32.7%)
MMM	0.76 (-44.7%)	0.52 (-19.2%)	**0.42**	0.52 (-19.2%)	0.6 (-30.0%)	0.59 (-28.8%)	0.6 (-30.0%)	0.61 (-31.1%)	1.14 (-63.2%)	1.14 (-63.2%)	1.12 (-62.5%)	27.99 (-98.5%)	1.0 (-58.0%)	0.88 (-52.3%)
MSFT	0.59 (-86.4%)	0.29 (-72.4%)	0.15 (-46.7%)	0.35 (-77.1%)	0.31 (-74.2%)	0.16 (-50.0%)	**0.08**	0.14 (-42.9%)	0.17 (-52.9%)	0.6 (-86.7%)	0.17 (-52.9%)	6.63 (-98.8%)	0.46 (-82.6%)	0.44 (-81.8%)
NKE	0.6 (-73.3%)	0.18 (-11.1%)	**0.16**	0.32 (-50.0%)	0.35 (-54.3%)	0.27 (-40.7%)	0.26 (-38.5%)	0.22 (-27.3%)	0.37 (-56.8%)	0.41 (-61.0%)	0.37 (-56.8%)	1.73 (-90.8%)	0.66 (-75.8%)	0.73 (-78.1%)
PG	1.24 (-35.5%)	0.94 (-14.9%)	0.92 (-13.0%)	1.08 (-25.9%)	0.92 (-13.0%)	**0.8**	0.81 (-1.2%)	0.91 (-12.1%)	1.3 (-38.5%)	1.16 (-31.0%)	1.31 (-38.9%)	9.66 (-91.7%)	1.23 (-35.0%)	1.04 (-23.1%)
UNH	0.64 (-53.1%)	0.38 (-21.1%)	**0.3**	**0.3**	**0.3**	0.42 (-28.6%)	0.38 (-21.1%)	0.36 (-16.7%)	0.42 (-28.6%)	0.43 (-30.2%)	0.42 (-28.6%)	4.94 (-93.9%)	0.59 (-49.2%)	0.61 (-50.8%)
V	0.49 (-63.3%)	0.41 (-56.1%)	0.34 (-47.1%)	0.29 (-37.9%)	0.37 (-51.4%)	0.34 (-47.1%)	0.27 (-33.3%)	**0.18**	0.61 (-70.5%)	1.01 (-82.2%)	0.61 (-70.5%)	1.38 (-87.0%)	0.56 (-67.9%)	0.56 (-67.9%)
VZ	**0.51**	0.78 (-34.6%)	0.65 (-21.5%)	0.63 (-19.0%)	0.67 (-23.9%)	0.84 (-39.3%)	0.57 (-10.5%)	0.7 (-27.1%)	0.59 (-13.6%)	0.9 (-43.3%)	0.59 (-13.6%)	12.4 (-95.9%)	0.94 (-45.7%)	0.58 (-12.1%)
WMT	0.94 (-59.6%)	0.39 (-2.6%)	**0.38**	0.52 (-26.9%)	0.44 (-13.6%)	0.42 (-9.5%)	0.46 (-17.4%)	0.47 (-19.1%)	0.45 (-15.6%)	0.68 (-44.1%)	0.43 (-11.6%)	0.56 (-32.1%)	0.84 (-54.8%)	0.76 (-50.0%)

Table 4.3: MASE loss comparison of individual stock symbols during non-volatile extrapolation period 2009-11 to 2019-11. SeTT & r-SeTT variants of WETT outperform the TFT variants most of the time.

ticker	WETT.a.all.sw	WETT.a.rsett.mw	WETT.a.sett.mw	WETT.a.tft.mw	WETT.q.all.sw	WETT.q.rsett.mw	WETT.q.sett.mw	WETT.q.tft.mw	arima	deepar	garch	nhits	rnn	tft
AAPL	**0.53**	0.77 (-31.2%)	0.85 (-37.6%)	0.63 (-15.9%)	0.8 (-33.8%)	0.98 (-45.9%)	0.99 (-46.5%)	1.66 (-68.1%)	1.2 (-55.8%)	2.06 (-74.3%)	1.23 (-56.9%)	1.58 (-66.5%)	1.56 (-66.0%)	1.48 (-64.2%)
AMGN	0.5 (-26.0%)	0.38 (-2.6%)	0.38 (-2.6%)	**0.37**	0.41 (-9.8%)	0.41 (-9.8%)	0.42 (-11.9%)	1.28 (-71.1%)	0.86 (-57.0%)	1.34 (-72.4%)	0.86 (-57.0%)	7.11 (-94.8%)	1.41 (-73.8%)	1.17 (-68.4%)
BA	0.98 (-41.8%)	0.81 (-29.6%)	0.66 (-13.6%)	0.71 (-19.7%)	0.58 (-1.7%)	0.69 (-17.4%)	0.69 (-17.4%)	**0.57**	1.44 (-60.4%)	1.19 (-52.1%)	1.4 (-59.3%)	5.59 (-89.8%)	2.02 (-71.8%)	1.47 (-61.2%)
CAT	0.74 (-29.7%)	0.56 (-7.1%)	**0.52**	0.59 (-11.9%)	0.59 (-11.9%)	0.55 (-5.5%)	0.58 (-10.3%)	0.73 (-28.8%)	0.97 (-46.4%)	0.65 (-20.0%)	0.94 (-44.7%)	7.4 (-93.0%)	0.76 (-31.6%)	0.99 (-47.5%)
CRM	**0.4**	0.75 (-46.7%)	0.81 (-50.6%)	0.63 (-36.5%)	0.87 (-54.0%)	1.04 (-61.5%)	1.06 (-62.3%)	1.46 (-72.6%)	1.58 (-74.7%)	1.47 (-72.8%)	1.6 (-75.0%)	2.6 (-84.6%)	1.49 (-73.2%)	0.45 (-11.1%)
CSCO	0.43 (-27.9%)	0.34 (-8.8%)	0.36 (-13.9%)	**0.31**	0.38 (-18.4%)	0.5 (-38.0%)	0.44 (-29.5%)	1.17 (-73.5%)	0.64 (-51.6%)	1.09 (-71.6%)	0.64 (-51.6%)	14.07 (-97.8%)	1.17 (-73.5%)	0.52 (-40.4%)
CVX	0.55 (-63.6%)	0.29 (-31.0%)	0.29 (-31.0%)	0.4 (-50.0%)	0.33 (-39.4%)	**0.2**	0.28 (-28.6%)	1.11 (-82.0%)	0.8 (-75.0%)	0.61 (-67.2%)	0.79 (-74.7%)	2.49 (-92.0%)	0.5 (-60.0%)	0.48 (-58.3%)
IBM	0.72 (-52.8%)	0.42 (-19.0%)	0.38 (-10.5%)	0.47 (-27.7%)	0.36 (-5.6%)	0.45 (-24.4%)	**0.34**	0.99 (-65.7%)	0.81 (-58.0%)	1.04 (-67.3%)	0.82 (-58.5%)	7.31 (-95.3%)	1.05 (-67.6%)	1.0 (-66.0%)
INTC	0.63 (-15.9%)	0.56 (-5.4%)	0.6 (-11.7%)	**0.53**	0.57 (-7.0%)	0.65 (-18.5%)	0.67 (-20.9%)	1.12 (-52.7%)	0.9 (-41.1%)	0.81 (-34.6%)	0.9 (-41.1%)	6.38 (-91.7%)	1.12 (-52.7%)	1.29 (-58.9%)
JNJ	0.54 (-48.1%)	0.29 (-3.4%)	0.32 (-12.5%)	**0.28**	0.34 (-17.6%)	0.46 (-39.1%)	0.41 (-31.7%)	1.85 (-84.9%)	0.67 (-58.2%)	0.94 (-70.2%)	0.67 (-58.2%)	0.71 (-60.6%)	1.02 (-72.5%)	2.33 (-88.0%)
KO	0.7 (-35.7%)	**0.45**	0.46 (-2.2%)	0.51 (-11.8%)	0.46 (-2.2%)	0.55 (-18.2%)	0.53 (-15.1%)	2.01 (-77.6%)	0.52 (-13.5%)	1.8 (-75.0%)	0.52 (-13.5%)	5.22 (-91.4%)	1.97 (-77.2%)	2.64 (-83.0%)
MCD	0.78 (-56.4%)	0.38 (-10.5%)	0.37 (-8.1%)	0.52 (-34.6%)	0.41 (-17.1%)	**0.34**	0.44 (-22.7%)	1.27 (-73.2%)	0.6 (-43.3%)	0.95 (-64.2%)	0.59 (-42.4%)	8.34 (-95.9%)	1.1 (-69.1%)	2.23 (-84.8%)
MMM	0.38 (-7.9%)	**0.35**	0.44 (-20.5%)	0.44 (-20.5%)	0.51 (-31.4%)	0.47 (-25.5%)	0.51 (-31.4%)	1.65 (-78.8%)	0.6 (-41.7%)	1.44 (-75.7%)	0.61 (-42.6%)	3.93 (-91.1%)	1.38 (-74.6%)	1.99 (-82.4%)
MSFT	**0.63**	0.84 (-25.0%)	0.92 (-31.5%)	0.77 (-18.2%)	0.92 (-31.5%)	0.98 (-35.7%)	0.98 (-35.7%)	1.35 (-53.3%)	1.73 (-63.6%)	1.63 (-61.4%)	1.73 (-63.6%)	1.39 (-54.7%)	1.0 (-37.0%)	1.5 (-58.0%)
NKE	1.02 (-28.4%)	0.88 (-17.0%)	0.87 (-16.1%)	0.9 (-18.9%)	0.79 (-7.6%)	**0.73**	0.75 (-2.7%)	0.94 (-22.3%)	1.57 (-53.5%)	1.22 (-40.2%)	1.56 (-53.2%)	3.44 (-78.8%)	1.28 (-43.0%)	1.04 (-29.8%)
PG	0.74 (-36.5%)	0.54 (-13.0%)	0.48 (-2.1%)	0.58 (-19.0%)	0.56 (-16.1%)	0.58 (-19.0%)	**0.47**	1.99 (-76.4%)	0.94 (-50.0%)	1.0 (-53.0%)	0.94 (-50.0%)	10.12 (-95.4%)	1.12 (-58.0%)	1.73 (-72.8%)
UNH	**0.21**	0.43 (-51.2%)	0.35 (-40.0%)	0.3 (-30.0%)	0.34 (-38.2%)	0.52 (-59.6%)	0.49 (-57.1%)	1.4 (-85.0%)	0.59 (-64.4%)	0.59 (-64.4%)	0.59 (-64.4%)	9.5 (-97.8%)	0.81 (-74.1%)	1.18 (-82.2%)
V	0.63 (-11.1%)	**0.56**	0.62 (-9.7%)	0.57 (-1.8%)	0.58 (-3.4%)	0.61 (-8.2%)	0.63 (-11.1%)	1.55 (-63.9%)	1.02 (-45.1%)	1.69 (-66.9%)	1.01 (-44.6%)	8.46 (-93.4%)	1.97 (-71.6%)	1.18 (-52.5%)
VZ	0.63 (-42.9%)	**0.36**	0.43 (-16.3%)	0.44 (-18.2%)	0.48 (-25.0%)	0.57 (-36.8%)	0.54 (-33.3%)	1.15 (-68.7%)	0.47 (-23.4%)	1.1 (-67.3%)	0.48 (-25.0%)	7.28 (-95.1%)	1.08 (-66.7%)	2.08 (-82.7%)
WMT	0.41 (-68.3%)	0.32 (-59.4%)	0.27 (-51.9%)	**0.13**	0.42 (-69.0%)	0.45 (-71.1%)	0.46 (-71.7%)	1.55 (-91.6%)	0.4 (-67.5%)	0.92 (-85.9%)	0.4 (-67.5%)	1.38 (-90.6%)	0.92 (-85.9%)	1.65 (-92.1%)

Table 4.4: MASE loss comparison of individual stock symbols during volatile extrapolation period 2009-11 to 2022-11. SeTT & r-SeTT variants of WETT outperform the TFT variants most of the time.

Ablation analysis and statistical test

Effect of the windowing ensemble on time series

To evaluate the effect of the windowing ensemble on the historical time series, we compared the result of the ensemble models with non-ensemble versions, based on the same models. Table 4.5 shows the 3 temporal Transformer models and their best-performing ensemble variants. Notably, the WETT-TFT variants demonstrate substantial improvements of more than 40% and 60% for the non-volatile and volatile periods, respectively. For all the WETT variants, we observe significant improvements over their equivalent non-ensemble models for both periods.

WETT ensembles can significantly improve the performance of temporal Transformer

start	end	yrs	condition	combination	model	duration (sec)	loss
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.rsett.mw	43.0	0.449 (-8.4%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.sett.mw	283.35	0.419 (-1.9%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.tft.mw	859.07	**0.411**
2009-11-01	2019-11-08	10	non-volatile	-	rsett_hd	210.91	0.713 (-42.4%)
2009-11-01	2019-11-08	10	non-volatile	-	sett_es	254.09	0.656 (-37.4%)
2009-11-01	2019-11-08	10	non-volatile	-	tft	193.2	0.701 (-41.4%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.rsett.mw	55.92	0.545 (-5.1%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.sett.mw	380.14	0.543 (-4.9%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.tft.mw	34.13	**0.517**
2009-11-01	2022-11-08	13	volatile	-	rsett_hd	225.91	0.585 (-11.7%)
2009-11-01	2022-11-08	13	volatile	-	sett_es	289.44	0.784 (-34.1%)
2009-11-01	2022-11-08	13	volatile	-	tft	202.12	1.368 (-62.2%)

Table 4.5: Comparison showing that WETT ensemble induces over 60% performance improvement on a temporal Transformer architecture, compared to when using the model without WETT ensemble. The WETT models are initialized with 3 different sets of weights.

models. In our experiments, a wett ensemble of temporal Transformer models outperformed a single temporal Transformer model by up to 60%.

Effect of weight diversification

For the created ensembles, we further assessed the effects of diversifying the base learners by using different seeds for initializing the weights. Recall that the suffixes *.sw* and *.mw* denote single-seed and multi-seed weight initialized variants of the WETT base learners, respectively. In Table 4.6, the results of each of the ensemble architectures and combination methods are paired by initialization variants, resulting in 6 each for the two market conditions.

In the non-volatile condition, the ensemble models initialized with multiple seeds consistently outperform their single-seed variant, across all six comparisons. However, the observation differs during the volatile extrapolation period, where an equal distribution is observed between the performance of the ensemble models initialized with single or multiple seeds across all pairs, as shown in Table 4.6. There is also a marginal difference of 0.4% between the best-performing single-seed weights (WETT.a.sett.sw) and multi-seed weights (WETT.a.tft.mw) ensemble models in this period, indicating that either initialization method is adequate during a volatile regime. Across the different market conditions, this supports our preference for multi-seed over single-seed initialization. In the best-case scenario, multi-seed weight variants demonstrate significantly better performance, while in

start	end	yrs	condition	combination	model	duration (sec)	loss
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.rsett.sw	357.79	0.533 (-23.0%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.rsett.mw	880.59	0.466 (-11.9%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.sett.sw	436.2	0.491 (-16.3%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.sett.mw	1120.14	0.429 (-4.3%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.tft.sw	347.39	0.525 (-21.7%)
2009-11-01	2019-11-08	10	non-volatile	q-stack	WETT.q.tft.mw	859.07	**0.411**
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.rsett.sw	15.78	0.463 (-11.2%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.rsett.mw	43.0	0.449 (-8.4%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.sett.sw	96.46	0.716 (-42.6%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.sett.mw	283.35	0.419 (-1.9%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.tft.sw	9.46	1.275 (-67.8%)
2009-11-01	2019-11-08	10	non-volatile	average	WETT.a.tft.mw	26.27	0.486 (-15.5%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.rsett.sw	421.09	0.573 (-10.2%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.rsett.mw	1140.77	0.616 (-16.4%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.sett.sw	529.47	0.651 (-21.0%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.sett.mw	1455.4	0.618 (-16.7%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.tft.sw	410.85	0.675 (-23.8%)
2009-11-01	2022-11-08	13	volatile	q-stack	WETT.q.tft.mw	1107.62	1.331 (-61.3%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.rsett.sw	19.83	0.594 (-13.4%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.rsett.mw	55.92	0.545 (-5.5%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.sett.sw	127.03	**0.514**
2009-11-01	2022-11-08	13	volatile	average	WETT.a.sett.mw	380.14	0.543 (-5.3%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.tft.sw	12.1	0.729 (-29.4%)
2009-11-01	2022-11-08	13	volatile	average	WETT.a.tft.mw	34.13	0.517 (-0.4%)

Table 4.6: Comparison of WETT models initialized with single and multiple seeds. The results indicate that when using the same base learner architecture, initializing the weights with 3 different seeds consistently outperforms initializing with a single seed during the non-volatile period. Both approaches perform equally during the volatile period.

the worst-case scenario, they perform on par with single-seed variants.

Statistical test

Finally, we test for the distinction between the error distribution of the different models using the non-parametric Mann–Whitney–Wilcoxon test (Paquet and Soleymani, 2022). The use of the Mann-Whitney-Wilcoxon test allows for the comparison of the error distributions without making assumptions about the underlying distributions. Table 4.7 compares

non-volatile period pair	t-test	volatile period pair	t-test
WETT.a.sett.mw/nhits	0.000708	WETT.a.sett.mw/nhits	0.000002
WETT.a.sett.mw/tft	0.000006	WETT.a.sett.mw/tft	0.000013
WETT.a.sett.mw/rnn	0.000006	WETT.a.sett.mw/rnn	0.000002
WETT.a.sett.mw/deepar	0.000006	WETT.a.sett.mw/deepar	0.000002
WETT.q.tft.mw/nhits	0.000483	WETT.q.tft.mw/nhits	0.000082
WETT.q.tft.mw/tft	0.000010	WETT.q.tft.mw/tft	0.595819
WETT.q.tft.mw/rnn	0.000004	WETT.q.tft.mw/rnn	0.216167
WETT.q.tft.mw/deepar	0.000006	WETT.q.tft.mw/deepar	0.153646
WETT.a.rsett.mw/nhits	0.000708	WETT.a.rsett.mw/nhits	0.000002
WETT.a.rsett.mw/tft	0.000395	WETT.a.rsett.mw/tft	0.000013
WETT.a.rsett.mw/rnn	0.000027	WETT.a.rsett.mw/rnn	0.000002
WETT.a.rsett.mw/deepar	0.000019	WETT.a.rsett.mw/deepar	0.000002
WETT.q.sett.mw/nhits	0.000851	WETT.q.sett.mw/nhits	0.000002
WETT.q.sett.mw/tft	0.000002	WETT.q.sett.mw/tft	0.000048
WETT.q.sett.mw/rnn	0.000002	WETT.q.sett.mw/rnn	0.000002
WETT.q.sett.mw/deepar	0.000002	WETT.q.sett.mw/deepar	0.000002
WETT.a.tft.mw/nhits	0.002325	WETT.a.tft.mw/nhits	0.000002
WETT.a.tft.mw/tft	0.000210	WETT.a.tft.mw/tft	0.000010
WETT.a.tft.mw/rnn	0.000063	WETT.a.tft.mw/rnn	0.000002
WETT.a.tft.mw/deepar	0.000134	WETT.a.tft.mw/deepar	0.000002
WETT.q.rsett.mw/nhits	0.001986	WETT.q.rsett.mw/nhits	0.000002
WETT.q.rsett.mw/tft	0.000105	WETT.q.rsett.mw/tft	0.000063
WETT.q.rsett.mw/rnn	0.000006	WETT.q.rsett.mw/rnn	0.000002
WETT.q.rsett.mw/deepar	0.000010	WETT.q.rsett.mw/deepar	0.000002

Table 4.7: Comparison of error distribution between the ensemble models and the comparison models using a non-parametric Mann–Whitney test. The table shows that the test statistics are less than 5% for 48 of the 51 comparisons, suggesting that the results are statistically significant.

the loss error of the top ensemble models with the other baseline models. The tests suggest that the null hypothesis is rejected, indicating that the results are unlikely to have occurred by chance, and our models have significantly improved predictive performance for most of the comparison. Of the 48 pairs in the comparison, only 3 pairs that compare the WETT-TFT-q-stack (WETT.q.tft.mw) with TFT, RNN & DeepAR during volatile period failed to reject the null hypothesis. For this period, we showed in Table 4.2 that the average combination variant is superior, as it demonstrates more than 60% improvement over TFT, RNN & DeepAR models.

4.4 Summary

In this chapter, we presented an ensemble of temporal Transformers models to effectively utilize an extended history of market data for financial time series forecasts. Our ensemble technique leverages a combination of base models generated from sliding windows of historical timeframes, with additional weight initialization diversification options for a complete experimentation regime. The base models are pre-trained, across a long historical period, broken down into smaller timeframes. We demonstrated the effectiveness of our method on a real-world financial dataset, outperforming several state-of-the-art baseline models. We used 20 symbols from the DJIA index and static metadata of the industry associated with each symbol to evaluate the performance in volatile and non-volatile market conditions on a 5-day multi-horizon forecast, following (Lim, Arik, et al., 2021).

Recall from Chapter 1 that there are three stages that financial market practitioners employ when applying machine learning to the stock market, namely data/feature engineering, price forecast, and investment strategy such as portfolio analysis (Chan, 2016). The scope of the work presented in this chapter is focused on price forecasts. With sufficiently improved price prediction algorithms, practitioners can apply such improvements to different strategies for meaningful market returns. Our results indicate that using sliding windows provides sufficient diversity for the ensemble model, thereby improving the predictive performance, without needing to combine multiple base models with different architectures. However, we aim to ensure that our models are not considered another black-box machine learning model. To this end, in the next chapter, we will incorporate explainability into the model development process to augment our comprehension of the temporal Transformer model while also improving on its predictive performance.

Chapter 5

Enhancing temporal Transformers via explainable surrogate

This chapter represents our research paper, namely:

- *Enhancing temporal Transformers for financial time series via local surrogate interpretability* that has been submitted for publication (Olorunnimbe and Viktor, 2024) (Olorunnimbe and Viktor, 2024a).
- In this research, we introduce a novel approach that leverages the interpretability of local surrogate models to enhance complex temporal Transformer models in financial forecasting. Utilizing insights from the surrogate models, we iteratively improve the model's predictive performance in both volatile and stable market conditions through informed feature selection.

5.1 Overview

In Chapter 2, we itemized research questions related to deep learning in the financial domain and summarized some works of research answering those questions. We prioritized works that have been backtested, suggesting that some due diligence has been done regarding the practicality and reproducibility of the research work. While acknowledging that more attention has been given to this area of study in recent years, it is clear that numerous research challenges remain. Based on the findings, we introduce temporal Transformer-based algorithms for financial forecast in Chapter 3, and extends this work

to ensembles in Chapter 4. In this chapter, we introduce the use of surrogate models to access explainability.

5.2 Local surrogate model for temporal Transformers

Deep learning is considered to still be in its infancy in financial applications, as it is considered a *black box*. However, numerous works have been conducted over the last few years to improve upon that notion. Notably, the Institute for Ethical AI & machine learning (EAIML), a UK-based think-tank devoted to the ethics in machine learning research with practical use cases, developed eight principles for responsible machine learning development (The Institute for Ethical AI & Machine Learning, 2020). It includes pertinent topics such as explainability, reproducibility, and practical accuracy. In recent times, attention has been given largely to the issue of XAI and Reproducible AI (Gundersen et al., 2018).

Recall that the Transformer model is fundamentally characterized by its attention mechanism, which computes the output as a weighted sum of values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Indeed, it is composed of Q, K, and V matrices, consisting of queries (q), keys (k), and values (v) vectors respectively.The q vector represents the information being queried, the k vector represents the information being compared against, and the v vector represents the query result to be generated (Russell and Norvig, 2021). In a time series scenario, the k vectors are generated from the historical data (e.g. the last 2 years), and the q vectors are generated from the input data (e.g. the last 30 days), representing the questions the model is asking about the historical data. The v vectors are the answers to those questions (i.e. the prediction horizon). In a TFT, a LSTM sequence-to-sequence layer is used to establish short-term relationships within the encoder and decoder, while the self-attention network is used for long-term dependencies (Lim, Arik, et al., 2021). Recall that SeTT and r-SeTT are further specialized for the financial domain by incorporating characteristics of financial time series into their architectures. This involves adapting the attention mechanism to reflect the similarity of time series sections to the most recent window, employing ES for SeTT and hd for r-SeTT, as similarity functions. The reader is referred back to Chapter 3 for more details.

To satisfy the need for model interpretability and explainability (i.e., XAI) beyond what is possible with the attention mechanism, we use a *surrogate model*. Surrogate models are simpler, interpretable models that are used to approximate and interpret the predictions of complex models (Molnar, 2022). There are two main types of surrogate models: global

surrogates, which provide an overall understanding by approximating the entire model behavior; and local surrogates, which focus on explaining individual predictions or specific instances. In this work, we employ local surrogate models to interpret the decisions made by the complex temporal Transformer model in specific financial time series. Local surrogates provide insights at individual prediction points (Ribeiro et al., 2016), which are essential for understanding the dynamic financial environments. Unlike global surrogates that offer a generalized perspective, local surrogates reveal specific, localized behaviors at the prediction points where data distributions or volatility may diverge from the general patterns. We use time series multivariate and univariate local explanations (TS-MULE) (Schlegel et al., 2021), an extension of local interpretable model-agnostic explanations (LIME) (Ribeiro et al., 2016), a framework for interpreting black-box models by approximating them locally with interpretable surrogate models. Similar to LIME, TS-MULE is model-agnostic, and it focuses on providing local explanations for individual predictions, enhancing transparency and trust in complex models. The primary idea is to *perturb* (generate) new samples around the input to be explained, obtain predictions for these samples from the black-box model, and then learn a simple model on this new dataset. By using the temporal aware TS-MULE, we address the fundamental temporal dependencies in time series data, which is not a consideration in LIME (Schlegel et al., 2021).

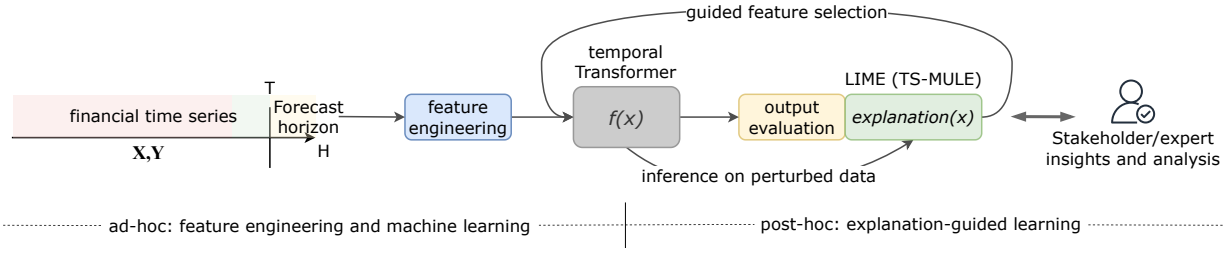


Figure 5.1: An illustration of the process flow for financial forecasting with explanation-guided learning. The feature engineering and machine learning steps are followed by explainability feedback to update the model after analysis from stakeholders.

Figure 5.1 presents an end-to-end approach for augmenting the interpretability and performance of the temporal Transformer model via EGL. Initially, feature engineering and exploratory data analysis (EDA) is conducted on financial time series data, establishing relevant features for the temporal Transformer to generate forecasts over a specified horizon. However, the influence of these features on model's performance cannot be determined until the EGL loop's post-hoc analysis is conducted. As noted above, we employ TS-MULE, as it is designed to address the temporal structure associated with time series data, unlike LIME that perturbs features independently, making the generated explanations less reliable for

time-dependent data in its default form. TS-MULE works in the following steps:

- Data segmentation: It first segments time series into interpretable components using segmentation techniques specific to time series. It employs novel strategies, including uniform, exponential, slope, and bin segmentation, to divide the time series into meaningful segments, either of equal length, increasing size, or based on local trends and patterns of related data regions, known as superpixels (Schlegel et al., 2021).

- Perturbation of segments: These segments are perturbed using zeros, inverse, or mean values as replacement strategies. This step aims to observe how changes influence the forecasting model's predictions, thereby identifying impactful data parts.

- Creation of surrogate model: A simpler, interpretable surrogate model is trained using the perturbed data to approximate the complex forecasting model's behavior. The coefficients of the simple model provide local explanations for the model's predictions, enabling explanations into the complex model's decision-making (Schlegel et al., 2021). As the simple interpretable model, we use Ridge regression, a form of regularized linear regression that addresses the potential for overfitting, a common concern in time series analysis due to autocorrelation and multicollinearity among variables (James et al., 2023). By introducing a regularization term, $l2$ norm, Ridge regression imposes a penalty on the size of coefficients, thus discouraging overfitting on the noise in the training data at the expense of generalizability (James et al., 2023).

- Generate explanation: The coefficient generated from the segments is reverted to time series data points and is subsequently visualized to provide insight into the effect of the individual features on the performance of the Transformer model. The process culminates in interpreting the surrogate model to identify and quantify the influence of specific time series segments on the model's predictions.

The generated explanations are integral to the post-hoc stage, where they are analyzed and used in a guided feature selection feedback loop to iteratively refine the model's performance. Algorithm 5.1 depicts the pseudocode of this EGL guided process.

Algorithm 5.1 Enhanced temporal Transformer using EGL

Input:
X, y ▷ financial time series data
Output: $\hat{y}$ ▷ enhanced H future forecasts

$X_{\mathrm{f}} \leftarrow$ FeatureConstruction(X) ▷ feature engineering and analysis
repeat
  $\mathtt{m} \leftarrow$ TrainTemporalTransformer(X_f, y)
  $\mathcal{E} \leftarrow$ GenerateExplaination$(\mathtt{m}, X_{\mathrm{f}})$ ▷ generate explanation
  $X_{\mathrm{f}} \leftarrow$ PostHocEGL$(\mathcal{E}, X_{\mathrm{f}})$
  $\mathtt{m} \leftarrow$ TrainTemporalTransformer(X_{f}, y)
until satisfied

procedure GenerateExplaination$(X_{\mathrm{f}}, \mathtt{m})$ ▷ via TS-MULE
  $\mathtt{m}_s \leftarrow$ SurrogateModel ▷ initialize surrogate model
  $\mathcal{C} \leftarrow \{\}$ ▷ initialize array for coefficients of surrogate model
  $\mathtt{S} \leftarrow$ SegmentTimeSeries(X_f) ▷ divide X_f into segments
  for each $\mathtt{s}$ in $\mathtt{S}$ **do**
    $\mathtt{s}_p \leftarrow$ PerturbSegment$(\mathtt{s})$ ▷ Generate perturbed segment data
    $\mathcal{C} \leftarrow \mathcal{C} \cup$ Explain$(\mathtt{s}_p, \mathtt{m}_s, \mathtt{m})$
  $\mathcal{E} \leftarrow$ ToSeries$(\mathcal{C})$ ▷ Convert segment coefficients to time series
  return $\mathcal{E}$

procedure PostHocEGL$(\mathcal{E}, X_{\mathrm{f}})$
  $\mathtt{Insights} \leftarrow$ Analyze$(\mathcal{E})$ ▷ expert stakeholder analysis of E
  $X_f \leftarrow$ RefineFeatures$(X_f, \mathtt{Insights})$ ▷ Update features based on insights
  return X_f

5.3 Experimental evaluation

Recall from Section 3.3.1, where we introduced a class of Transformer models with temporal features, namely TFT, SeTT and r-SeTT. Our experiment consists of evaluating the predictive performance of these temporal Transformer models by comparing the quantile loss results from before the XAI-based interactive machine learning step with those from after its implementation.

5.3.1 Experimentation setup

For the temporal Transformer models, TFT, SeTT and r-SeTT, we used the following hyperparameters based on our experimentation in Chapter 3: learning rate of 0.018, dropout rate of 0.3, and current window of 30 days. Also, we used ES and hd for the similarity function of the base learner SeTT and r-SeTT base learners respectively. For TS-MULE, we employed Ridge as the interpretable model, using the default settings of slope segmentation and zero values as the replacement strategy for perturbation within the segments.

Data

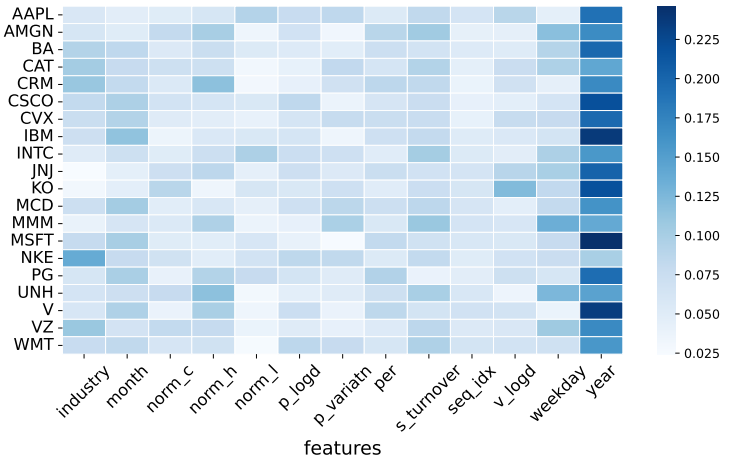
We used time series stock market and the financial data presented in Section 3.4.1. to evaluate our model. As expected, features such as the normalized high, low, and closing prices (`norm_h`, `norm_l`, and `norm_c` respectively) are identified as strongly correlated during our EDA, because they are all normalized with the opening stock price (see Table 3.2). `Industry` and `ticker` are weakly correlated with other features in the financial time series dataset. The multicollinearity observed among these features is typically less problematic for deep learning models due to their ability to extract complex patterns and the effectiveness of regularization techniques such as dropout in mitigating overfitting concerns (Srivastava et al., 2014).

We constructed independent data sets consisting of features of the 20 companies, across two sets of independent periods, one volatile and the other non-volatile, based on the VIX. The volatile periods, chosen for their high market volatility as indicated by the VIX, were from February 1, 2016, to February 1, 2018, and from February 1, 2015, to February 1, 2018. The non-volatile periods, characterized by lower VIX values, were from January 1, 2016, to January 1, 2018, and from January 1, 2015, to January 1, 2018. Each period comprises 2-year and 3-year intervals respectively. Our selection of these historical timeframes is informed by our experimentation in Chapter 3, which indicates that such periods yield more optimal results. All but the last 5 trading days of each period were used for training, with the model subsequently tested on the final 5 days, constituting the test set.

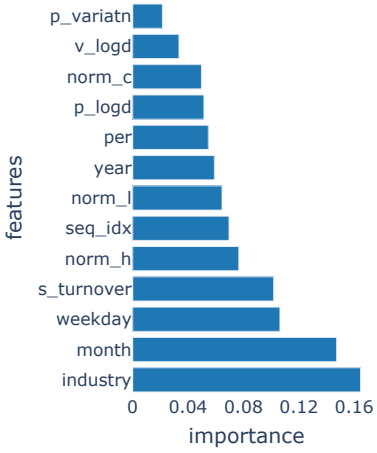
5.3.2 Results and discussion

In this section, we provide the details of the outcome of our experiments, conducted over four distinct periods. Following the initial training step of each period, we evaluate the

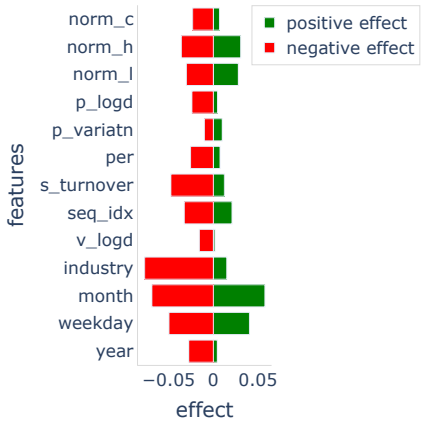
importance and influence of the input features generated by TS-MULE. The corresponding feature importance and effect plots for the volatile market conditions are shown in Figure 5.2.



(a) Feature importance for r-SeTT - Feb. 1, 2016, to Feb. 1, 2018.



(b) Average feature importance.

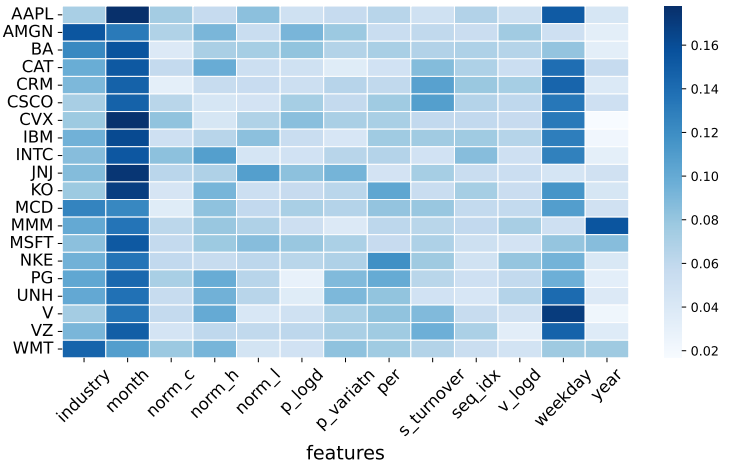


(c) Average effect of the features.

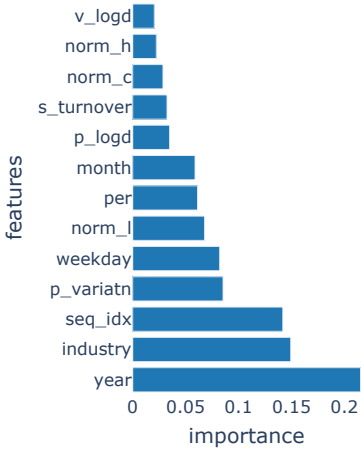
Figure 5.2: Pre-EGL: Features importance and effects on the model predictions across all 20 symbols during volatile extrapolation periods before the explanation-guided features selection process.

Figure 5.3 presents the feature importance and their effects during non-volatile market

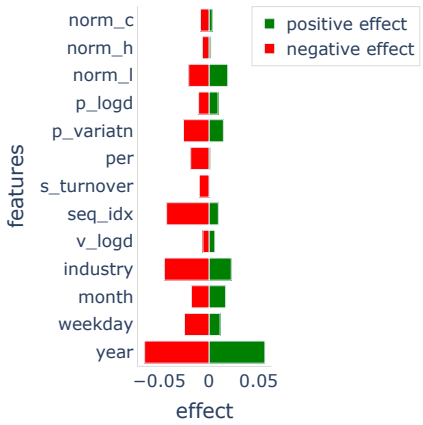
conditions. As in the case of volatile periods, the features mostly have a negative impact, although the extent of their influence varies. Furthermore, the effect of the trading volume (`v_logd`) is comparatively minor relative to the other features.



(a) Feature importance for r-SeTT - Jan. 1, 2016 to Jan. 1, 2018.



(b) Average feature importance.



(c) Average effect of the features.

Figure 5.3: Pre-EGL: Features importance and effects on the model predictions across all 20 symbols during non-volatile extrapolation periods before the explanation-guided features selection process.

Following the insights gained from this analysis, we adjusted the input features for

the models and conducted a subsequent round of modeling. The outcomes of this revised modeling are displayed in Figures 5.5 and 5.4.

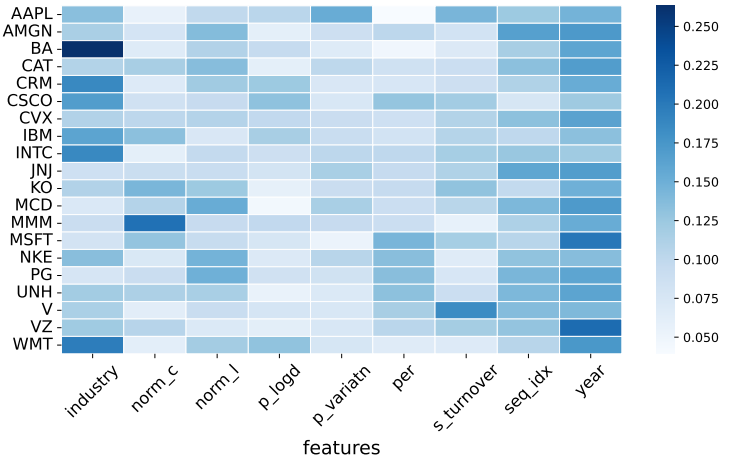
During the volatile extrapolation period, Figure 5.2(a) shows that the initial training of the r-SeTT model disproportionately focused on the `year` feature, overshadowing other critical features like normalized close (`norm_c`) or changes in trading volume (`v_logd`). However, the combined perspectives in Figure 5.2(b,c) indicate a more balanced, yet still significant, emphasis on date-related features, as evidenced by the feature importance and effects plots, based on the output of TS-MULE. These features primarily exhibit a negative influence. While the direction of the effect (negative or positive) is not inherently indicative of quality, the prevalent focus on these features warrants consideration.

Figure 5.4 demonstrates improvements in the model's performance during non-volatile periods after updating the input features following the post-EGL phase. Notably, the `industry` feature continues to be prominent, as depicted in Figure 5.4(b); this feature was not modified due to the lack of suitable alternatives in the model to represent the industry aspects of market symbols. Furthermore, Figure 5.4(c) indicates that the influence of the `industry` feature on the model's predictions is neither overwhelmingly positive nor negative.

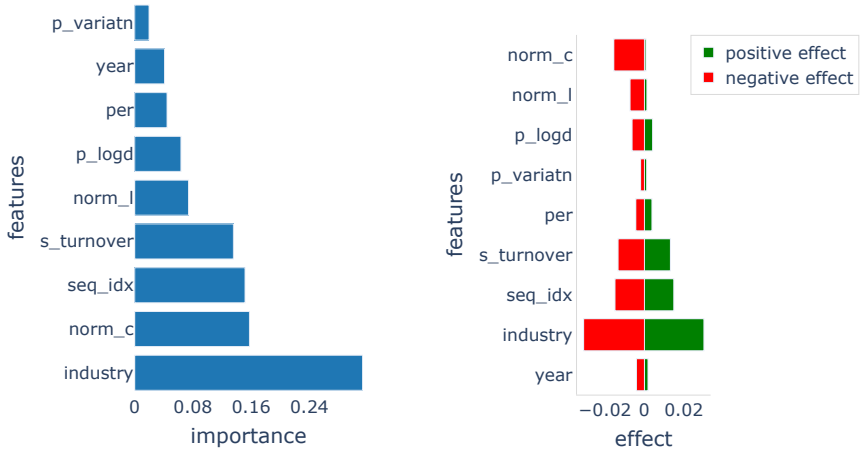
Informed by our findings, we eliminated the excessively weighted date-related features, and `norm_c` for volatile periods. The rationale for removing `norm_c` is that its primary information, the closing price, is already captured in other features like `p_logd` and `p_variatn`. For non-volatile periods, which exhibit a distinct relevance profile, we discarded the two least significant features, `norm_h` and `v_logd`. From the EDA step, we know that `norm_h` is highly correlation with other normalized price features, and `v_logd`'s information regarding volume is sufficiently represented by `s_turnover` (volume relative to shares outstanding), which holds greater feature importance. Additionally, we opted to remove `month` and `weekday` features. However, we retained the `year` feature in this context, as preliminary tests indicate its significant negative impact on results when absent.

Following the explanation-guided feature selection, the model is re-trained using the updated input features. Upon conducting a post-EGL explanability, we observed notable improvements in the model's focus on the updated input features, as indicated by Figures 5.5. The feature representation in the r-SeTT models has significantly improved, particularly for the volatile period, which is evident from the altered feature importance in the average plot. Improvements in feature relevance for the non-volatile period have also been observed. However, further analysis is necessary to conclusively determine their generalizable impact on performance.

Table 5.1 presents the loss results, contrasting the performance before and after the



(a) Feature importance for r-SeTT - Jan. 1, 2016 to Jan. 1, 2018.

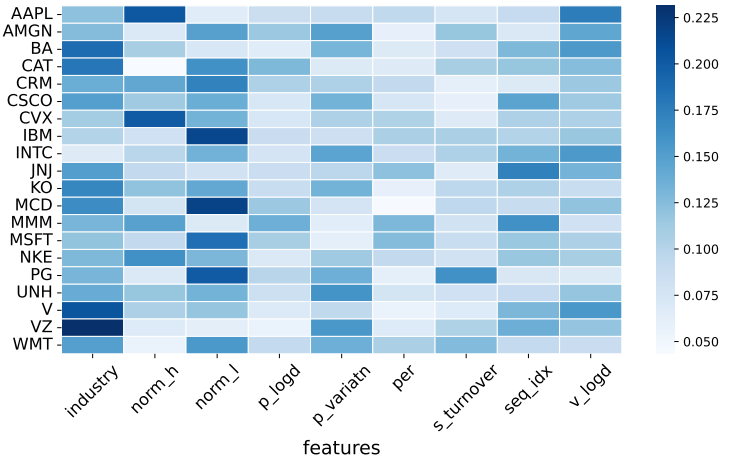


(b) Average feature importance.

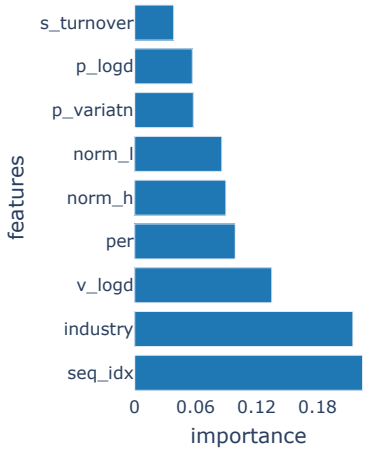
(c) Average effect of the features.

Figure 5.4: Post-EGL: Features importance and effects on the model predictions across all 20 symbols during non-volatile extrapolation periods after the explanation-guided features selection process.

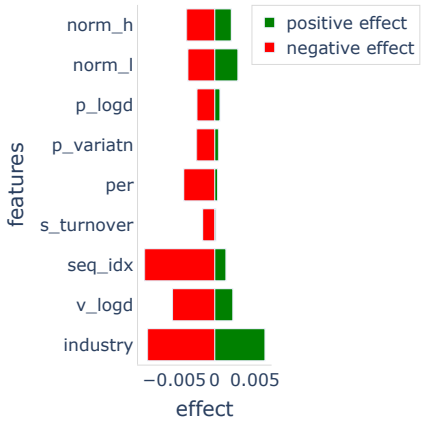
explanation-guided model updates. We employed the MASE metric to assess the performance of each of the three models, comparing their pre and post-guided learning phases for every period examined. This approach enabled us to demonstrate significant improvements in performance using the same Transformer architecture and input features. This itera-



(a) Feature importance for r-SeTT - Feb. 1, 2016, to Feb. 1, 2018.



(b) Average feature importance.



(c) Average effect of the features.

Figure 5.5: Post-EGL: Features importance and effects on the model predictions across all 20 symbols during volatile extrapolation periods after the explanation guided features selection process.

tive improvement is not possible using EDA-based feature engineering. The influence of XAI in directing us towards varied input features for different periods further underscores this point. The table reveals a consistent improvement in the loss metrics for the volatile extrapolation period, ranging from 11% to 24%. While the improvements on three of the

start	end	yrs	model	loss	start	end	yrs	model	loss
2016-02-01	2018-02-01	2	rsett_hd	1.19 (-21.7%)	2016-01-01	2018-01-01	2	rsett_hd	**0.523**
2016-02-01	2018-02-01	2	rsett_hd (post)	**0.932**	2016-01-01	2018-01-01	2	rsett_hd (post)	0.554 (-5.5%)
2015-02-01	2018-02-01	3	rsett_hd	0.922 (-31.2%)	2015-01-01	2018-01-01	3	rsett_hd	0.468 (-8.3%)
2015-02-01	2018-02-01	3	rsett_hd (post)	**0.634**	2015-01-01	2018-01-01	3	rsett_hd (post)	**0.43**
2016-02-01	2018-02-01	2	sett_es	1.064 (-13.3%)	2016-01-01	2018-01-01	2	sett_es	**0.508**
2016-02-01	2018-02-01	2	sett_es (post)	**0.922**	2016-01-01	2018-01-01	2	sett_es (post)	0.542 (-6.3%)
2015-02-01	2018-02-01	3	sett_es	0.949 (-21.9%)	2015-01-01	2018-01-01	3	sett_es	**0.436**
2015-02-01	2018-02-01	3	sett_es (post)	**0.742**	2015-01-01	2018-01-01	3	sett_es (post)	0.446 (-2.3%)
2016-02-01	2018-02-01	2	tft	1.068 (-11.3%)	2016-01-01	2018-01-01	2	tft	0.502 (-7.5%)
2016-02-01	2018-02-01	2	tft (post)	**0.948**	2016-01-01	2018-01-01	2	tft (post)	**0.465**
2015-02-01	2018-02-01	3	tft	0.924 (-23.9%)	2015-01-01	2018-01-01	3	tft	0.472 (-3.7%)
2015-02-01	2018-02-01	3	tft (post)	**0.702**	2015-01-01	2018-01-01	3	tft (post)	**0.455**

Volatile extrapolation period | Non-volatile extrapolation period

Table 5.1: Loss metric of six pairs of volatile and non-volatile extrapolation periods for two and three years timeframe.

six non-volatile extrapolation periods are modest, ranging from 3% to 8%, they are still noteworthy. Furthermore, the extent of deterioration in performance for the remaining three comparisons in this period is relatively lower, lying within the 2-6% range.

Discussion

Our experimental results underscore the significance of integrating XAI into the interactive machine learning pipeline for Transformer models. Despite thorough feature engineering before training, incorporating XAI into our methodology has enabled a clearer understanding of the impact of data features on prediction outcomes. Table 5.1 illustrates that as a result of the explainability feedback, we achieve substantial performance improvements, with gains up to 24% in volatile extrapolation periods after the EGL phase, regardless of market complexity. We, the researchers, acted as the domain experts to analyze the explanation and further refine the features in this experiment.

Moreover, the insights gained from this interpretability are valuable for market analysts without a background in machine learning, enabling them to understand the specific factors driving the model's predictions and offer suggestions for improving performance. This is particularly pertinent when considering that such analysts often possess a deeper understanding of stock market fundamentals than a typical machine learning engineer or

researcher. This methodology also lays the groundwork for a more comprehensive optimization strategy. It goes beyond adjusting the input hyperparameters of the machine learning model, but also learning from the XAI outputs to retrain and further enhance the model's performance in a guided reinforcing loop.

5.4 Summary

In this chapter, we introduce an approach that uses local surrogate models to enhance the predictive performance of temporal Transformer models across different financial market conditions through insights gleaned from a post-training XAI step. Our experimental evaluations spanned 2-year and 3-year timeframes, utilizing three distinct temporal Transformer models. The results notably indicate marked improvements in forecasting accuracy during volatile market periods, affirming the effectiveness of our approach and yielding promising outcomes in non-volatile periods as well. This research not only validates the integration of explainability into financial modeling but also establishes a promising direction for future financial forecasting methodologies. It signals a shift towards more robust, intuitive, and reliable forecasting models in the financial domain. In the next chapter, we conclude the thesis and introduce potential future avenues of research.

Chapter 6

Conclusion and future work

As deep learning becomes more common in financial research, attention is being increasingly focused on ensuring that the research process conforms to procedures established in the financial domain. A recent example of this is the renewed attention on backtesting algorithms using historical data and domain-specific financial evaluation metrics, as we show in our survey in Chapter 2. Historically, a limiting factor for deep learning in the mainstream application is the need for higher compute requirements, but graphics processing unit (GPU) and neural processors have become ubiquitous for various machine learning use cases, both in cloud and non-cloud-based computing resources. This has made computing-intensive algorithms more attractive, as the time it takes to train models is no longer considered a major limiting factor. This shift is already accelerating the application of deep learning in real-world financial scenarios, spurred by its superior predictive capabilities, and practical applications, as demonstrated by LLMs.

Recall from Chapter 1 that the application of machine learning in the financial market involves data/feature engineering, price forecasts, and investment strategy. It is common to see real-world use cases across all three stages depending on traditional statistical and machine-learning approaches because of a need for explainability and ease of use. We introduce algorithms and propose approaches that cut across these stages, in order to conduct holistic research in practical deep learning involving financial time series data.

6.1 Contributions

The contributions of this thesis are as follows.

6.1.1 Temporal Transformers with financial data characteristics

We introduced the similarity embedded temporal Transformer (SeTT) and run-similarity embedded temporal Transformer (r-SeTT) architectures shown to enhance financial forecasting by integrating the heteroscedasticity financial time series properties (i.e., non-constant volatility over time) into temporal Transformers. The models constructed excel beyond traditional and baseline models by embedding a similarity vector into the attention mechanism. This vector mutes dissimilar time series during training, significantly improving prediction accuracy across multiple timeframes. By focusing on similar historical data patterns, these architectures address the inherent volatility and non-constant variance of financial markets, making deep learning models more adaptable and effective for financial forecasting. This innovative approach offers a focused learning process, leading to notable improvements in predictive performance and demonstrating the potential for specialized Transformers in financial market analysis.

6.1.2 Extensive timeframe analysis for financial market data

We conducted a comprehensive analysis across various timeframes to showcase the robustness and generalizability of the SeTT and r-SeTT models, demonstrating consistent performance across a wide range of market conditions and timeframes. Despite the optimization of comparison models' hyperparameters, our models outperformed others during volatile periods and maintained competitiveness during non-volatile periods. Performance during volatile periods is critical, as they represent moments of uncertainty and opportunity, with models able to effectively navigate these conditions will significantly bolster investment strategies and mitigate risk (R. D. Edwards et al., 2021). This was attributed to the models' ability to focus on timeframes most similar to the current window, leveraging the similarity vector for better extrapolations. The study provides strong evidence of the effectiveness of our approach in predicting stock trends, highlighting its potential value as a tool in financial forecasting.

6.1.3 Efficient data utilization via ensemble of temporal Transformers

An ensemble approach named windowing ensemble of temporal Transformers (WETT) was introduced in Chapter 4, which significantly improves predictive performance and computational efficiency for extended timeframes exceeding 10 years, in comparison to single-model

Transformer approaches. By combining multiple temporal Transformer models learned within sliding windows, the ensemble efficiently utilizes available data over an extended timeframe. We demonstrated the ensemble model's construction from sliding windows of historical financial time series data, its pre-training, and the use of a model registry for consistency and model reusability, underscoring the ensemble's improved predictive performance and computational efficiency. The ensemble approach, through its combination methods and diversity of constituent models, offers better generalization to unseen data compared to single-model strategies, showcasing advantages in complex processes such as deep learning, where individual models may be susceptible to overfitting.

6.1.4 Enhanced predictability via surrogate models

We introduced an approach that uses local surrogate models to enhance the predictive performance of temporal Transformer models across different financial market conditions through insights gleaned from a post-training XAI step. The explanation-guided learning (EGL) framework employs a cyclical process where explanations generated from the local surrogate models, guided by TS-MULE, inform iterative refinements in feature selection and model training. This process starts with feature construction, followed by model training and explanation generation. The explanations are then analyzed by domain experts to refine features iteratively until satisfaction is achieved. This EGL-guided process not only augments model performance by focusing on features with significant predictive power but also enriches model interpretability, making complex forecasting models more transparent and trustworthy. Experimental evaluations demonstrate that this approach leads to marked improvements in forecasting accuracy across different financial market conditions, underscoring the efficacy of integrating explainability into the financial modeling process.

6.2 Future work

6.2.1 Temporal Transformers with industry-wide insights

Expanding upon the SeTT, r-SeTT and WETT architectures, our future work will extend its methodology to include a wider scope of similarity analysis, encompassing both industry-wide and general stock market trends. Additionally, we plan to refine our approach to similarity by introducing probabilistic measures, or, degrees of similarity derived from the comparison between windows. Such an approach would allow us to move beyond

binary evaluations to capture more nuanced variations in relatedness. This will involve broadening the lookup window to incorporate analyses of related companies and trends across sectors, aiming to significantly enhance the predictive accuracy and efficacy of deep learning models in financial forecasting. This expansion will leverage interconnected market dynamics, potentially revealing more profound insights into the predictive relationships between related companies and overall market movements, thus offering a richer, more nuanced understanding of market behaviors and trends.

6.2.2 Multimodal learning with temporal Transformers

Recall that in Chapter 2, we discussed various financial data types: market, fundamental, and alternate data. So far, we used a combination of market and fundamental data due to their numeric nature and direct compatibility with deep learning models. However, a significant amount of data exists in various formats such as news articles, academic publications, social media posts, and more, encompassing text, images, audio, and video (Ridhawi and Osman, 2023). We propose employing multimodal learning with temporal Transformers to incorporate these diverse data forms. This enriched approach simulates complex market environments by synthesizing heterogeneous information sources. This comprehensive approach offers a richer view of market dynamics by capturing a wider spectrum of influencing factors across multiple sources. Future efforts will focus on fine-tuning this integration, promising to elevate financial forecasting's precision and offer a more comprehensive market perspective.

6.2.3 EGL-guided refinement with expert feedback

In Chapter 5, we showed how using a local surrogate model can improve the performance of a temporal Transformer model through expert intervention. For future work, we will investigate a hybrid EGL framework that incorporates an XAI feedback loop to automate most of the intervention, with integration into the model optimization process. Such a system should be designed to complement expert judgment by systematically identifying input features that either exhibit a high correlation with other features or have a disproportionate impact on feature importance, potentially skewing model interpretations, amongst other relevant rules. The expert will then review and exclude the top n features from the next training step. This integration with the optimization algorithm, alongside periodic expert reviews, ensures a dynamic refinement process based on both insights generated from XAI and human expertise. This hybrid collaboration should be able to streamline

the refinement of predictive models by ensuring that only the most relevant and non-redundant features contribute to the model's decision-making process. Consequently, such an approach may significantly boost model accuracy through optimized feature selection and enhance interpretability. Emphasizing expert feedback within this EGL-guided process encourages the development of transparent, trustworthy models essential in domains where prediction justification is as crucial as accuracy, such as financial forecasting. The integration of XAI with human feedback into model development represents a significant step forward in the quest for more efficient, scalable, accurate, and interpretable predictive models.

6.2.4 Deep reinforcement learning based investment strategy using temporal Transformers

Recall that the final step in applying machine learning to financial markets focuses on developing an investment strategy. With the performance and explainability enhancements achieved so far in this thesis, along with future recommendations for further enhancements, our objective is to support the formulation of an investment strategy whose profitability can be assessed using financial metrics. Building on our exploration of deep learning applications in financial markets, we propose advancing to a reinforcement learning model that employs multi-horizon quantile regression outputs from our improved temporal Transformer models to formulate a deep reinforcement learning DRL-based investment strategy.

The multi-horizon quantile forecasts generated by our WETT ensemble model serve as inputs for a reinforcement learning framework dedicated to financial investment strategies. Here, DRL evaluates risk management decisions or trading actions based on these inputs. The advantage of quantile forecasting lies in its ability to present both best-case and worst-case scenarios for the prediction target across a probabilistic spectrum, aiding in the risk optimization of financial decisions (Lim, Arik, et al., 2021). The actor-critic method (i.e., a DRL approach where the "actor" proposes actions and the "critic" evaluates them for future reward potential) in our proposed financial investment strategy would utilize multi-horizon quantile forecasts from temporal Transformer models. Literature has shown that actor-critic reinforcement agents, encompassing the advantages of actor-only and critic-only agents, are very successful in complex environments such as the financial market (Soleymani and Paquet, 2021; S. Sun et al., 2023). As part of the preliminary experimentation for future work, we will compare the actor-critic with other agents for completeness.

The actor component would make investment decisions (buy, hold, sell) based on these forecasts, while the critic would evaluate these decisions by estimating future returns,

incorporating risk management. This dynamic would allow the strategy to adapt to market changes, optimizing for profitability and risk. In alignment with financial sector practices and as detailed in Chapter 2, performance will be assessed through backtesting on historical financial data and against various financial metrics, including cumulative wealth and the Sharpe ratio, to evaluate effectiveness and risk management.

References

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016). "Tensorflow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283.

Aceto, Giuseppe, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescape (2019). "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges". eng. In: *IEEE eTransactions on network and service management* 16.2, pp. 445–458. ISSN: 1932-4537.

Adosoglou, George, Gianfranco Lombardo, and Panos M. Pardalos (2020). "Neural network embeddings on corporate annual filings for portfolio selection". In: *Expert Systems with Applications*. DOI: 10.1016/j.eswa.2020.114053.

Ahrefs (2020). *Google search operators: The complete list (42 advanced operators)*. URL: ahrefs.com/blog/google-advanced-search-operators.

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama (July 2019). "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage: ACM, pp. 2623–2631. DOI: 10.1145/3292500.3330701.

Amel-Zadeh, Amir, Jan-Peter Calliess, Daniel Kaiser, and Stephen Roberts (January 2020). "Machine learning-based financial statement analysis". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3520684.

Arimond, Alexander, Damian Borth, Andreas G. F. Hoepner, Michael Klawunn, and Stefan Weisheit (June 2020). "Neural networks and value at risk". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3591996.

Armbrust, Michael, Ali Ghodsi, Reynold Xin, and Matei Zaharia (2021). "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics". In: *11th Conference on Innovative Data Systems Research (CIDR)*, p. 8. URL: researchr.org/publication/Zaharia0XA21.

Arnott, Robert D., Campbell R. Harvey, and Harry Markowitz (November 2018). "A backtesting protocol in the era of machine learning". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3275654.

Bacon, Carl R (January 2023). *Practical portfolio performance measurement and attribution*. 3rd edition. Wiley finance. Newark: Wiley. ISBN: 978-1-119-83196-9.

Baek, Yujin and Ha Young Kim (2018). "ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module". In: *Expert Systems with Applications*. DOI: 10.1016/j.eswa.2018.07.019.

Bao, Wenhang and Xiao-yang Liu (June 2019). "Multi agent deep reinforcement learning for liquidation strategy analysis". In: *arXiv e-prints*. DOI: 10.48550/arxiv.1906.11046. eprint: 1906.11046.

Bengio, Y. (November 1997). "Using a financial training criterion rather than a prediction criterion." In: *International journal of neural systems* 8.4, pp. 433–443. DOI: 10.1142/s0129065797000422.

Bergmeir, Christoph and José M. Benítez (May 2012). "On the use of cross-validation for time series predictor evaluation". In: *Information Sciences* 191, pp. 192–213. ISSN: 00200255.

Boedihardjo, Horatio, Xi Geng, Terry Lyons, and Danyu Yang (April 2016). "The signature of a rough path: Uniqueness". In: *Advances in Mathematics* 293, pp. 720–737. DOI: 10.1016/j.aim.2016.02.011.

Brocke, Jan vom, Alan Hevner, and Alexander Maedche (January 2020). *Design science research. Cases*. 1st edition. Springer. 327 pp. DOI: 10.1007/978-3-030-46781-4.

Brockwell, Peter J. and Richard A. Davis (2016). *Introduction to time series and forecasting*. Springer Texts in Statistics. New York: Springer International Publishing. DOI: 10.1007/978-3-319-29854-2.

Buehler, Hans, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood (June 2020). "A data-driven market simulator for small data environments". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3632431.

Castro, Leandro Nunes de (June 2006). *Fundamentals of natural computing: Basic concepts, algorithms, and applications*. 1st edition. Chapman and Hall/CRC. 696 pp. ISBN: 978-1138581944.

Chakole, Jagdish and Manish Kurhekar (August 2020). "Trend following deep Q-learning strategy for stock trading". In: *Expert Systems* 37.4. DOI: 10.1111/exsy.12514.

Challu, Cristian, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski (June 2023). "NHiTS: Neural hierarchical interpolation for time series forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.6, pp. 6989–6997. DOI: 10.1609/aaai.v37i6.25854.

Chalvatzis, Chariton and Dimitrios Hristu-Varsakelis (2020). "High-performance stock index trading via neural networks and trees". In: *Applied Soft Computing* 96, p. 106567. DOI: 10.1016/j.asoc.2020.106567.

Chan, Ernest P. (December 2016). *Machine trading: Deploying computer algorithms to conquer the markets*. 1st edition. Wiley. ISBN: 978-1119219606.

Chan, Ernest P. (November 2020). *Financial machine learning - A practitioner's perspective*. URL: youtube.com/watch?v=BhaJVZNpL4M.

Chan, Ernest P. (July 2021). *Quantitative trading: How to build your own algorithmic trading business*. 2nd edition. Hoboken, New Jersey: Wiley. 256 pp. ISBN: 978-1-119-80006-4.

Chen, Andrew, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Clemens Mewald, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Avesh Singh, Fen Xie, Matei Zaharia, Richard Zang, Juntai Zheng, and Corey Zumar (2020). "Developments in MLflow: A system to accelerate the machine learning lifecycle". In: *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. DEEM'20. Portland, OR, USA: Association for Computing Machinery. DOI: 10.1145/3399579.3399867.

Chen, Lin, Zhilin Qiao, Minggang Wang, Chao Wang, Ruijin Du, and Harry Eugene Stanley (July 2018). "Which artificial intelligence algorithm better predicts the chinese stock market?" In: *IEEE Access* 6, pp. 48625–48633. DOI: 10.1109/access.2018.2859809.

Chen, Yu Ying, Wei Lun Chen, and Szu Hao Huang (September 2018). "Developing arbitrage strategy in high-frequency pairs trading with filterbank cnn algorithm". In: *Proceedings - 2018 IEEE International Conference on Agents, ICA 2018*. Institute of Electrical and Electronics Engineers Inc., pp. 113–116. DOI: 10.1109/agents.2018.8459920.

Chong, Eunsuk, Chulwoo Han, and Frank C. Park (2017). "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies". In: *Expert Systems with Applications* 83, pp. 187–205. DOI: 10.1016/j.eswa.2017.04.030.

Chow, K Victor, Wanjun Jiang, and Jingrui Li (2021). "Does VIX truly measure return volatility?" In: *Handbook of Financial Econometrics, Mathematics, Statistics, and Machine Learning*. World Scientific, pp. 1533–1559. DOI: 10.2139/ssrn.2489345.

Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes (September 2017). "Supervised learning of universal sentence representations from natural language inference data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 670–680. DOI: 10.18653/v1/D17-1070.

Databricks (2022). *What is a medallion architecture?* Databricks. URL: databricks.com/glossary/medallion-architecture.

Davchev, Jovan, Kostadin Mishev, Irena Vodenska, Ljubomir Chitkushev, and Dimitar Trajanov (December 2020). "Bitcoin price prediction using transfer learning on financial micro-blogs". In: *OpenBU*. URL: hdl.handle.net/2144/43647.

Day, Min Yuh and Chia Chou Lee (November 2016). "Deep learning for financial sentiment analysis on finance news providers". In: *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016*. Institute of Electrical and Electronics Engineers Inc., pp. 1127–1134. DOI: 10.1109/asonam.2016.7752381.

Easley, David, Marcos M. López de Prado, and Maureen O'Hara (2012). "The volume clock: Insights into the high frequency paradigm". In: *The Journal of Portfolio Management* 39.1, pp. 19–29. DOI: 10.3905/jpm.2012.39.1.019.

Edwards, Robert D., John Magee, and W. H. C. Bassetti (December 2021). *Technical analysis of stock trends*. 11th edition. CRC Press. 686 pp. ISBN: 978-1-03-224182-1.

Edwards, Tim and Hamish Preston (2017). "A practitioner's guide to reading VIX". In: *S&P Global: S&P Dow Jones indices (Education, strategy 201)*, pp. 1–10. URL: spglobal.com/spdji/en/education-a-practitioners-guide-to-reading-vix.pdf.

Erlemann, Rasmus, Richard Lockhart, and Rihan Yao (2022). "Cramér-von Mises tests for change points". In: *Scandinavian journal of statistics* 49.2. DOI: 10.1111/sjos.12544.

Fabozzi, Frank J. and Marcos López De Prado (September 2018). "Being honest in backtest reporting: A template for disclosing multiple tests". In: *Journal of Portfolio Management* 45.1, pp. 141–147. DOI: 10.3905/jpm.2018.45.1.141.

Fang, Yujie, Juan Chen, and Zhengxuan Xue (2019). "Research on quantitative investment strategies based on deep learning". In: *Algorithms*. DOI: 10.3390/a12020035.

Ferguson, Ryan and Andrew Green (October 2018). "Deeply learning derivatives". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3244821.

Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan (June 2020). "Deep ensembles: A loss landscape perspective". In: *arXiv e-prints*. DOI: 10.48550/1912.02757.

François-Lavet, Vincent, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau (2018). "An introduction to deep reinforcement learning". In: *Foundations and Trends in Machine Learning*. DOI: 10.48550/1811.12560.

Franses, Philip Hans (January 2016). "A note on the mean absolute scaled error". In: *International Journal of Forecasting* 32.1. DOI: 10.1016/j.ijforecast.2015.03.008.

Ganaie, M.A., Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan (October 2022). "Ensemble deep learning: A review". In: *Eng. Appl. Artif. Intell.* 115.C. DOI: 10.1016/j.engappai.2022.105151.

Goerg, Sebastian J. and Johannes Kaiser (2009). "Nonparametric testing of distributions: The Epps-Singleton two-sample test using the empirical characteristic function". In: *The Stata Journal* 9.3, pp. 454–465. DOI: 10.1177/1536867x0900900307.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. Cambridge: MIT Press. ISBN: 978-0-262-03561-3.

Google (2020). *Google Scholar*. URL: scholar.google.ca.

Gundersen, Odd Erik, Y. Gil, and D. Aha (2018). "On reproducible AI: Towards reproducible research, open science, and digital scholarship in AI publications". In: *AI Magazine* 39, pp. 56–68. DOI: 10.1609/aimag.v39i3.2816.

Gunning, David and David W. Aha (2019). "DARPA's explainable artificial intelligence program". In: *The AI magazine* 40.2, pp. 44–58. DOI: 10.1609/aimag.v40i2.2850.

Guo, Yifeng, Xingyu Fu, Yuyan Shi, and Mingwen Liu (May 2018). "Robust log optimal strategy with reinforcement learning". In: *arXiv e-prints*. DOI: 10.48550/1805.00205.

Haibe-Kains, Benjamin, George Alexandru Adam, Ahmed Hosny, Farnoosh Khodakarami, Levi Waldron, Bo Wang, Chris McIntosh, Anna Goldenberg, Anshul Kundaje, Casey S. Greene, Tamara Broderick, Michael M. Hoffman, Jeffrey T. Leek, Keegan Korthauer, Wolfgang Huber, Alvis Brazma, Joelle Pineau, Robert Tibshirani, Trevor Hastie, John P. A. Ioannidis, John Quackenbush, and Hugo J. W. L. Aerts (October 2020). "Transparency and reproducibility in artificial intelligence". In: *Nature* 586.7829, E14–E16. DOI: 10.1038/s41586-020-2766-y.

Han, Jiawei, Micheline Kamber, and Jian Pei (2012). *Data mining: Concepts and techniques*. Elsevier Inc. DOI: 10.1016/c2009-0-61819-5.

Hinton, Geoffrey (2017). "Boltzmann machines". In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, pp. 164–168. DOI: 10.1007/978-1-4899-7687-1_31.

Hollander, Myles, Douglas A. Wolfe, and Eric Chicken (December 2013). *Nonparametric statistical methods*. 3rd edition. New Jersey: Wiley. 848 pp. ISBN: 978-0-470-38737-5.

Hu, Guosheng, Yuxin Hu, Kai Yang, Zehao Yu, Flood Sung, Zhihong Zhang, Fei Xie, Jianguo Liu, Neil Robertson, Timpathy Hospedales, and Qiangwei Miemie (September 2018). "Deep stock representation learning: From candlestick charts to investment decisions". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Vol. 2018. Institute of Electrical and Electronics Engineers Inc., pp. 2706–2710. DOI: 10.1109/icassp.2018.8462215.

Hu, Zexin, Yiqi Zhao, and Matloob Khushi (March 2021). "A survey of forex and stock price prediction using deep learning". In: *Applied System Innovation* 4.1, p. 9. DOI: 10.3390/asi4010009.

Hyndman, Rob and G. Athanasopoulos (2021). *Forecasting: Principles and practice*. 3rd edition. Melbourne, Australia: OTexts. ISBN: 978-0987507136.

Hyndman, Rob J. and Anne B. Koehler (October 2006). "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* 22.4, pp. 679–688. DOI: 10.1016/j.ijforecast.2006.03.001.

Insights, Deloitte (2019). *AI leaders in financial services*. URL: deloitte.com/us/en/insights/industry/financial-services/artificial-intelligence-ai-financial-services-frontrunners.html.

Institute, Corporate Finance (2020). *Backtesting - Overview, how it works, common measures*. en-US. URL: corporatefinanceinstitute.com/resources/knowledge/trading-investing/backtesting.

Investing.com (2013). *AAPL — Apple stock price*. URL: investing.com/equities/apple-computer-inc.

Ivanov, Sergey and Alexander D'yakonov (June 2019). "Modern deep reinforcement learning algorithms". In: *arXiv e-prints*. DOI: 10.48550/1906.10025.

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor (2023). *An introduction to statistical learning: With applications in Python*. Cham: Springer International Publishing. ISBN: 978-3-031-38746-3.

Koshiyama, Adriano, Stefano B. Blumberg, Nick Firoozye, Philip Treleaven, and Sebastian Flennerhag (April 2020). "QuantNet: Transferring learning across systematic trading strategies". In: *arXiv e-prints*. DOI: 10.48550/2004.03445.

Kusuma, Rosdyana Mangir Irawan, Trang-Thi Ho, Wei-Chun Kao, Yu-Yen Ou, and Kai-Lung Hua (February 2019). "Using deep learning neural networks and candlestick chart representation to predict stock market". In: *arXiv e-prints*. DOI: 10.48550/1903.12258.

Lei, Ying, Qinke Peng, and Yiqing Shen (April 2020). "Deep learning for algorithmic trading: Enhancing macd strategy". In: *ACM International Conference Proceeding Series*. New York, NY, USA: Association for Computing Machinery, pp. 51–57. DOI: 10.1145/3404555.3404604.

Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman (January 2020). *Mining of massive datasets*. 3rd edition. Cambridge: Cambridge University Press. 565 pp. ISBN: 978-1-108-47634-8.

Li, Shiyang, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan (December 2019). "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 471. New York: Curran Associates Inc., pp. 5243–5253. DOI: 10.48550/arxiv.1907.00235.

Li, Xinyi, Yinchuan Li, Yuancheng Zhan, and Xiao-Yang Liu (June 2019). "Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation". In: *arXiv e-prints*. DOI: 10.48550/arXiv.1907.01503.

Li, Yuming, Pin Ni, and Victor Chang (June 2020). "Application of deep reinforcement learning in stock trading strategies and stock forecasting". In: *Computing* 102.6, pp. 1305–1322. DOI: 10.1007/s00607-019-00773-w.

Liang, Zhipeng, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li (August 2018). "Adversarial deep reinforcement learning in portfolio management". In: *arXiv e-prints*. DOI: 10.48550/1808.09940.

Lim, Bryan, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister (2021). "Temporal fusion Transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting* 37.4, pp. 1748–1764. DOI: 10.1016/j.ijforecast.2021.03.012.

Lim, Bryan and Stefan Zohren (2021). "Time-series forecasting with deep learning: A survey". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379. DOI: 10.1098/rsta.2020.0209.

Lopez de Prado, Marcos (August 2013). "What to look for in a backtest". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.2308682.

Lu, Jie, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang (2018). "Learning under concept drift: A review". In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1. DOI: 10.1109/tkde.2018.2876857.

Luk, Martin (September 2023). "Generative AI: Overview, economic impact, and applications in asset management". In: *SSRN Electronic Journal* 4574814. DOI: 10.2139/ssrn.4574814.

Maeda, Iwao, David DeGraw, Michiharu Kitano, Hiroyasu Matsushima, Hiroki Sakaji, Kiyoshi Izumi, and Atsuo Kato (April 2020). "Deep reinforcement learning in agent based financial market simulation". In: *Journal of Risk and Financial Management* 13.4, p. 71. DOI: 10.3390/jrfm13040071.

Malkiel, Burton Gordon (2023). *A random walk down Wall Street: The time-tested strategy for successful investing*. 50th edition. New York. ISBN: 978-1-324-05113-8.

Mazur, Mieszko, Man Dang, and Miguel Vega (January 2021). "COVID-19 and the march 2020 stock market crash. Evidence from S&P1500". In: *Finance Research Letters* 38, p. 101690. DOI: 10.1016/j.frl.2020.101690.

Molnar, Christoph (2022). *Interpretable machine learning: A guide for making black box models explainable*. 2nd edition. ISBN: 9798411463330.

Montiel, Jacob, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet (2020). "River: Machine learning for streaming data in Python". In: *arXiv e-prints*. DOI: 10.48550/10.48550/2012.04740.

Montiel, Jacob, Jesse Read, Albert Bifet, and Talel Abdessalem (2018). "Scikit-multiflow: A multi-output streaming framework". In: *Journal of Machine Learning Research* 19.72, pp. 2915–2914. DOI: 10.5555/3291125.3309634.

Müller, Vincent C. (2020). "Ethics of artificial intelligence and robotics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Metaphysics Research Lab, Stanford University. URL: plato.stanford.edu/archives/win2020/entries/ethics-ai.

Mustapa, Farah Hayati and Mohd Tahir Ismail (November 2019). "Modelling and forecasting S&P 500 stock prices using hybrid Arima-Garch model". In: *Journal of Physics: Conference Series* 1366.1, p. 012130. DOI: 10.1088/1742-6596/1366/1/012130.

O'Shea, Timothy and Jakob Hoydis (2017). "An introduction to deep learning for the physical layer". In: *IEEE Transactions on Cognitive Communications and Networking* 3.4, pp. 563–575. DOI: 10.1109/tccn.2017.2758370.

Olorunnimbe, Kenniy and Herna L. Viktor (June 2022a). "Deep learning in the stock market—a systematic survey of practice, backtesting, and applications". In: *Artificial Intelligence Review* 56.3, pp. 2057–2109. DOI: 10.1007/s10462-022-10226-0.

Olorunnimbe, Kenniy and Herna L. Viktor (October 2022b). "Similarity embedded temporal Transformers: Enhancing stock predictions with historically similar trends". In: *26th International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pp. 388–398. DOI: 10.1007/978-3-031-16564-1_37.

Olorunnimbe, Kenniy and Herna L. Viktor (June 2024a). "Enhancing temporal Transformers for financial time series via local surrogate interpretability". Accepted for presentation in 27th International Symposium on Methodologies for Intelligent Systems (ISMIS).

Olorunnimbe, Kenniy and Herna L. Viktor (March 2024b). "Ensemble of temporal Transformers for financial time series". In: *Journal of Intelligent Information Systems*. DOI: 10.1007/s10844-024-00851-2.

Olorunnimbe, Kenniy and Herna L. Viktor (April 2024c). "Towards efficient similarity embedded temporal Transformers via extended timeframe analysis". In: *Complex & Intelligent Systems*. DOI: 10.1007/s40747-024-01400-8.

Ong, Eng-Jon and Miroslaw Bober (June 2016). "Improved hamming distance search using variable length hashing". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. NV: IEEE, pp. 2000–2008. DOI: 10.1109/cvpr.2016.220.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe (2022). "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems*. Vol. 35, pp. 27730–27744. DOI: 10.48550/arXiv.2203.02155.

Paquet, Eric and Farzan Soleymani (June 2022). "QuantumLeap: Hybrid quantum neural network for financial predictions". In: *Expert Systems with Applications* 195, p. 116583. DOI: 10.1016/j.eswa.2022.116583.

Park, Hyungjun, Min Kyu Sim, and Dong Gu Choi (2020). "An intelligent financial portfolio trading strategy using deep Q-learning". In: *Expert Systems with Applications*. DOI: 10.1016/j.eswa.2020.113573.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An imperative style, high-performance deep learning library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*. Red Hook, NY, USA: Curran Associates Inc. DOI: 10.5555/3454287.3455008.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830. URL: jmlr.org/papers/v12/pedregosa11a.html.

Pradeepkumar, Dadabada and Vadlamani Ravi (September 2017). "Forecasting financial time series volatility using Particle Swarm Optimization trained quantile regression neural network". In: *Applied Soft Computing* 58. ISSN: 15684946. DOI: 10.1016/j.asoc.2017.04.014.

Prado, Marcos Lopez de (February 2018). *Advances in financial machine learning*. 1st edition. New Jersey: Wiley. 400 pp. ISBN: 978-1-119-48208-6.

Raghubir, Priya and Sanjiv R. Das (April 2010). "The long and short of it: Why are stocks with shorter runs preferred?" In: *Journal of Consumer Research* 36.6, pp. 964–982. DOI: 10.1086/644762.

Raman, Natraj and Jochen L. Leidner (2019). "Financial market data simulation using deep intelligence agents". In: *Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection*. Ed. by Yves Demazeau, Eric Matson, Juan Manuel Corchado, and Fernando De la Prieta. Cham: Springer International Publishing, pp. 200–211. ISBN: 978-3-030-24209-1.

"Reporting standards and availability of data, materials, code and protocols" (2013). In: *Nature Portfolio*. DOI: 10.25504/fairsharing.hra50k.

Research data policy Springer Nature (2023). URL: springernature.com/gp/authors/research-data-policy.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (August 2016). ""Why should i trust you?": Explaining the predictions of any classifier". In: *Proceedings of the 22nd*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16. New York: Association for Computing Machinery, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

Ridhawi, Mohammad Al and Hussein Al Osman (June 2023). "Stock market prediction from sentiment and financial stock data using machine learning". In: *Proceedings of the Canadian Conference on Artificial Intelligence*. DOI: 10.21428/594757db.40c1a462.

Ruf, Johannes and Weiguan Wang (2020). "Hedging with linear regressions and neural networks". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3580132.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature*. DOI: 10.1038/323533a0.

Russell, Stuart and Peter Norvig (May 2021). *Artificial intelligence: A modern approach*. 4th global edition. Harlow: Pearson. 1168 pp. ISBN: 978-1-292-40113-3.

Salinas, David, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski (July 2020). "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3, pp. 1181–1191. DOI: 10.1016/j.ijforecast.2019.07.001.

Samek, Wojciech, Thomas Wiegand, and Klaus Robert Müller (August 2017). "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models". In: *arXiv e-prints*. DOI: 10.48550/1708.08296.

Schlegel, Udo, Duy Lam Vo, Daniel A. Keim, and Daniel Seebacher (2021). "TS-MULE: Local interpretable model-agnostic explanations for time series forecast models". In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Cham: Springer International Publishing, pp. 5–14. DOI: 10.1007/978-3-030-93736-2_1.

Scholar, Semantic (2020). *AI-powered research tool*. URL: semanticscholar.org.

Seese, Detlef, Christof Weinhardt, and Frank Schlottmann, eds. (June 2008). *Handbook on information technology in finance*. Berlin Heidelberg: Springer. 833 pp. ISBN: 978-3-540-49486-7.

Sezer, Omer Berat, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu (May 2020). "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019". In: *Applied Soft Computing* 90, p. 106181. DOI: 10.1016/j.asoc.2020.106181.

Silva, Thalita R., Audeliano W. Li, and Edson O. Pamplona (September 2020). "Automated trading system for stock index using lstm neural networks and risk management". In: *Proceedings - 2020 International Joint Conference onNeural Networks (IJCNN)*. Institute of Electrical and Electronics Engineers (IEEE), pp. 1–8. DOI: 10.1109/ijcnn48605.2020.9207278.

SimFin (2021). *Simplifying Finance*. URL: simfin.com.

Soleymani, Farzan and Eric Paquet (October 2020). "Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath". In: *Expert Systems with Applications* 156, p. 113456. DOI: 10.1016/j.eswa.2020.113456.

Soleymani, Farzan and Eric Paquet (2021). "Deep graph convolutional reinforcement learning for financial portfolio management – DeepPocket". In: *Expert Systems with Applications* 182. DOI: 10.1016/j.eswa.2021.115127.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (January 2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15.1, pp. 1929–1958. DOI: 10.5555/2627435.2670313.

Sun, Shuo, Rundong Wang, and Bo An (March 2023). "Reinforcement learning for quantitative trading". In: *ACM Trans. Intell. Syst. Technol.* 14.3. DOI: 10.1145/3582560.

Sun, Tong, Jia Wang, Jing Ni, Yu Cao, and Benyuan Liu (December 2019). "Predicting futures market movement using deep neural networks". In: *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 118–125. DOI: 10.1109/icmla.2019.00027.

Tay, Yi, Mostafa Dehghani, Dara Bahri, and Donald Metzler (April 2022). "Efficient Transformers: A survey". In: *ACM Computing Surveys*. DOI: 10.1145/3530811.

Taylor, James W. (July 2000). "A quantile regression neural network approach to estimating the conditional density of multiperiod returns". In: *Journal of Forecasting* 19.4. ISSN: 0277-6693, 1099-131x.

The Institute for Ethical AI & Machine Learning (2020). *The 8 principles for responsible development of AI & machine learning systems*. URL: ethical.institute/principles.html.

Théate, Thibaut and Damien Ernst (April 2020). "An application of deep reinforcement learning to algorithmic trading". In: *arXiv e-prints*. DOI: 10.48550/2004.06627.

Tornes, Adam and Leanne Trujillo (2021). *Enabling the future of academic research with the Twitter API*. URL: blog.twitter.com/developer/en%5C_us/topics/tools/2021/enabling-the-future-of-academic-research-with-the-twitter-api.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. DOI: 10.48550/arxiv.1706.03762.

Wachowicz, Erin (2020). "Wharton research data services (WRDS)". In: *Journal of Business and Finance Librarianship* 25.3-4, pp. 184–187. DOI: 10.1080/08963568.2020.1847552.

Wang, J., T. Sun, B. Liu, Y. Cao, and D. Wang (2018). "Financial markets prediction with deep learning". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 97–104. DOI: 10.1109/icmla.2018.00022.

Wang, Jia, Tong Sun, Benyuan Liu, Yu Cao, and Hongwei Zhu (2019). "CLVSA: A convolutional LSTM based variational sequence-to-sequence modelwith attention for predicting trends of financial markets". In: *IJCAI International Joint Conference on Artificial Intelligence*. Vol. 2019-8. International Joint Conferences on Artificial Intelligence, pp. 3705–3711. DOI: 10.24963/ijcai.2019/514.

Wang, Jifei and Lingjing Wang (October 2019). "Residual switching network for portfolio optimization". In: *arXiv e-prints*. DOI: 10.48550/1910.07564.

Wang, Jingyuan, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong (2019). "AlphaStock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, pp. 1900–1908. DOI: 10.1145/3292500.3330647.

Wang, Jun, Qian Yang, Zhanlei Jin, Wenzhang Chen, Tiejun Pan, and Jian Shen (April 2020). "Research on quantitative trading strategy based on LSTM". In: *Proceedings of 2020 Asia-Pacific Conference on Image Processing, Electronics and Computers, IPEC 2020*. Institute of Electrical and Electronics Engineers Inc., pp. 266–270. DOI: 10.1109/ipec49694.2020.9115114.

Wen, Min, Ping Li, Lingfei Zhang, and Yan Chen (2019). "Stock market trend prediction using high-order information of time series". In: *IEEE Access* 7, pp. 28299–28308. DOI: 10.1109/access.2019.2901842.

Wen, Ruofeng, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka (2017). "A multi-horizon quantile recurrent forecaster". In: *NeurIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*. New York: Curran Associates Inc. DOI: 10.48550/arxiv.1711.11053.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, and et al. (March 2016). "The FAIR guiding principles for scientific data management and stewardship". In: *Scientific Data* 3.1, pp. 1–9. DOI: 10.1038/sdata.2016.18.

Wojtas, Maksymilian and Ke Chen (2020). "Feature importance ranking for deep learning". In: *arXiv e-prints*. DOI: 10.48550/2010.08973.

Wu, Jia, Chen Wang, Lidong Xiong, and Hongyong Sun (July 2019). "Quantitative trading on stock market based on deep reinforcement learning". In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 2019-7. Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/ijcnn.2019.8851831.

Wu, Jimmy Ming Tai, Mu En Wu, Pang Jen Hung, Mohammad Mehedi Hassan, and Giancarlo Fortino (October 2020). "Convert index trading to option strategies via LSTM

architecture". In: *Neural Computing and Applications*, pp. 1–18. DOI: 10.1007/s00521-020-05377-6.

Xiao, Cao (2021). *Introduction to deep learning for healthcare*. eng. Cham, Switzerland: Springer. ISBN: 9783030821845.

Yang, Junming, Yaoqi Li, Xuanyu Chen, Jiahang Cao, and Kangkang Jiang (November 2019). "Deep learning for stock selection based on high frequency price volume data". In: *arXiv e-prints*. DOI: 10.48550/1911.02502.

Yang, Li and Abdallah Shami (2020). "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415, pp. 295–316. DOI: 10.1016/j.neucom.2020.07.061.

Yang, Steve Y., Yangyang Yu, and Saud Almahdi (2018). "An investor sentiment reward-based trading system using Gaussian inverse reinforcement learning algorithm". In: *Expert Systems with Applications*. DOI: 10.1016/j.eswa.2018.07.056.

Yoo, Jaemin, Yejun Soun, Yong-chan Park, and U Kang (August 2021). "Accurate multivariate stock movement prediction via data-axis transformer with multi level contexts". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. New York: Association for Computing Machinery. DOI: 10.1145/3447548.3467297.

Zaharia, Matei, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica (2010). "Spark: Cluster computing with working sets". In: *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. Boston, MA, p. 10. DOI: 10.5555/1863103.1863113.

Zhang, Chuheng, Yuanqi Li, Xi Chen, Yifei Jin, Pingzhong Tang, and Jian Li (October 2020). "DoubleEnsemble: A new ensemble method based on sample reweighting and feature selection for financial data analysis". In: *arXiv e-prints*. DOI: 10.48550/2010.01265.

Zhang, Haiying, Qiaomei Liang, Siheng Li, Rongqi Wang, and Qingqiang Wu (June 2020). "Research on stock prediction model based on deep learning". In: *Journal of Physics: Conference Series* 1549, p. 022124. DOI: 10.1088/1742-6596/1549/2/022124.

Zhang, Haiying, Qiaomei Liang, Rongqi Wang, and Qingqiang Wu (September 2020). "Stacked model with autoencoder for financial time series prediction". In: *15th International Conference on Computer Science and Education, ICCSE 2020*. Institute of Electrical and Electronics Engineers (IEEE), pp. 222–226. DOI: 10.1109/iccse49874.2020.9201745.

Zhang, Zihao, Stefan Zohren, and Stephen Roberts (June 2019). "DeepLOB: Deep convolutional neural networks for limit order books". In: *IEEE Transactions on Signal Processing* 67.11, pp. 3001–3012. DOI: 10.1109/tsp.2019.2907260.

Zhang, Zihao, Stefan Zohren, and Stephen Roberts (April 2020). "Deep reinforcement learning for trading". In: *The Journal of Financial Data Science* 2.2, pp. 25–40. DOI: 10.3905/jfds.2020.1.030.

Zhao, Ran, Yuntian Deng, Mark Dredze, Arun Verma, David Rosenberg, and Amanda Stent (September 2018). "Visual attention model for cross sectional stock return prediction and end to end multimodal market representation learning". In: *arXiv e-prints*. DOI: 10.48550/1809.03684.

Appendix A

Hyperparameters of comparison between deep learning models

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
1	non-volatile	-1.0548	35	0.0	0.006	deepar	0
1	non-volatile	0.0804	2	0.0	0.001	tft	16
1	non-volatile	0.0146	42	0.0	0.034	rnn	0
1	non-volatile	0.0512	32	0.0	0.001	nhits	0
1	non-volatile	-1.2719	44	0.0	0.032	deepar	0
1	non-volatile	0.0159	39	0.0	0.039	rnn	0
1	non-volatile	0.1045	1	0.4	0.038	tft	30
1	non-volatile	0.0615	41	0.0	0.001	nhits	0
1	non-volatile	0.065	45	0.0	0.0	nhits	0
1	non-volatile	-0.9585	21	0.0	0.004	deepar	0
1	non-volatile	0.0285	42	0.0	0.056	rnn	0
1	non-volatile	0.0829	11	0.600	0.793	tft	30
1	non-volatile	-0.7871	43	0.0	0.045	deepar	0
1	non-volatile	0.03	39	0.0	0.052	rnn	0
1	non-volatile	0.0716	3	0.3	0.037	tft	30
1	non-volatile	0.0807	5	0.0	0.001	nhits	0
1	non-volatile	0.1091	21	0.0	0.004	nhits	0
1	non-volatile	0.0271	39	0.0	0.049	rnn	0
1	non-volatile	-0.0293	22	0.0	0.01	deepar	0
1	non-volatile	0.0959	0	0.3	0.082	tft	30
2	non-volatile	0.0722	4	0.2	0.001	tft	30
2	non-volatile	0.0176	48	0.0	0.084	rnn	0
2	non-volatile	-1.1507	21	0.0	0.004	deepar	0
2	non-volatile	0.0553	35	0.0	0.0	nhits	0

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
2	non-volatile	0.066	3	0.3	0.037	tft	30
2	non-volatile	-0.5202	40	0.0	0.013	deepar	0
2	non-volatile	0.0724	5	0.0	0.001	nhits	0
2	non-volatile	0.0266	22	0.0	0.015	rnn	0
2	non-volatile	0.077	13	0.2	0.315	tft	30
2	non-volatile	-1.8998	44	0.0	0.018	deepar	0
2	non-volatile	0.0614	22	0.0	0.0	nhits	0
2	non-volatile	0.0197	24	0.0	0.027	rnn	0
2	non-volatile	0.0228	42	0.0	0.029	rnn	0
2	non-volatile	-1.2925	37	0.0	0.02	deepar	0
2	non-volatile	0.0661	1	0.4	0.038	tft	30
2	non-volatile	0.1005	22	0.0	0.0	nhits	0
2	non-volatile	-0.0245	41	0.0	0.011	deepar	0
2	non-volatile	0.1501	28	0.0	0.002	nhits	0
2	non-volatile	0.0762	35	0.4	0.009	tft	30
2	non-volatile	0.0359	39	0.0	0.049	rnn	0
3	non-volatile	0.0256	25	0.0	0.018	rnn	0
3	non-volatile	0.0957	5	0.0	0.001	nhits	0
3	non-volatile	0.0566	4	0.2	0.001	tft	30
3	non-volatile	-0.3923	29	0.0	0.009	deepar	0
3	non-volatile	0.0161	32	0.0	0.051	rnn	0
3	non-volatile	-1.2576	27	0.0	0.014	deepar	0
3	non-volatile	0.0956	47	0.0	0.0	nhits	0
3	non-volatile	0.0816	4	0.2	0.001	tft	30
3	non-volatile	0.0799	43	0.0	0.0	nhits	0
3	non-volatile	-0.4596	40	0.0	0.102	deepar	0
3	non-volatile	0.0162	25	0.0	0.019	rnn	0
3	non-volatile	0.0745	0	0.3	0.082	tft	30
3	non-volatile	0.0183	43	0.0	0.056	rnn	0
3	non-volatile	0.0888	27	0.0	0.0	nhits	0
3	non-volatile	0.0605	1	0.4	0.038	tft	30
3	non-volatile	-0.1673	48	0.1	0.067	deepar	0
3	non-volatile	0.0331	33	0.0	0.067	rnn	0
3	non-volatile	0.4716	18	0.1	0.021	deepar	0
3	non-volatile	0.0799	21	0.2	0.001	tft	30
3	non-volatile	0.2053	22	0.0	0.002	nhits	0
5	non-volatile	-1.0567	27	0.0	0.033	deepar	0
5	non-volatile	0.1467	27	0.0	0.0	nhits	0
5	non-volatile	0.0187	41	0.0	0.05	rnn	0
5	non-volatile	0.0729	10	0.6	0.0	tft	30
5	non-volatile	0.1068	3	0.3	0.037	tft	30
5	non-volatile	-0.2474	36	0.0	0.007	deepar	0
5	non-volatile	0.0911	23	0.0	0.0	nhits	0
5	non-volatile	0.0508	26	0.0	0.039	rnn	0

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
5	non-volatile	0.068	21	0.2	0.001	tft	30
5	non-volatile	-0.8271	34	0.1	0.02	deepar	0
5	non-volatile	0.0165	44	0.0	0.051	rnn	0
5	non-volatile	0.0915	27	0.0	0.0	nhits	0
8	non-volatile	-1.2994	13	0.0	0.006	deepar	0
8	non-volatile	0.0215	25	0.0	0.019	rnn	0
8	non-volatile	0.1639	22	0.0	0.0	nhits	0
8	non-volatile	0.0593	29	0.3	0.063	tft	30
8	non-volatile	0.0221	41	0.0	0.082	rnn	0
8	non-volatile	-1.3149	25	0.0	0.026	deepar	0
8	non-volatile	0.1394	41	0.0	0.0	nhits	0
8	non-volatile	0.1019	43	0.9	0.012	tft	30
8	non-volatile	-0.7062	19	0.1	0.029	deepar	0
8	non-volatile	0.0667	16	0.1	0.943	tft	30
8	non-volatile	0.1657	44	0.0	0.0	nhits	0
8	non-volatile	0.0147	42	0.0	0.05	rnn	0
13	volatile	0.2177	49	0.0	0.001	nhits	0
13	volatile	0.0661	32	0.0	0.015	rnn	0
13	volatile	0.102	1	0.4	0.038	tft	30
13	volatile	-0.4961	39	0.0	0.006	deepar	0
1	volatile	0.076	46	0.0	0.002	nhits	0
1	volatile	-0.6419	37	0.0	0.045	deepar	0
1	volatile	0.0508	39	0.0	0.052	rnn	0
1	volatile	0.0657	3	0.3	0.037	tft	30
1	volatile	0.0792	3	0.3	0.037	tft	30
1	volatile	-0.8274	27	0.0	0.018	deepar	0
1	volatile	0.017	30	0.0	0.016	rnn	0
1	volatile	0.0543	21	0.0	0.004	nhits	0
1	volatile	0.054	14	0.1	0.005	tft	30
1	volatile	0.0937	24	0.0	0.002	nhits	0
1	volatile	0.0182	41	0.0	0.027	rnn	0
1	volatile	-1.1668	43	0.0	0.016	deepar	0
1	volatile	0.0382	35	0.0	0.083	rnn	0
1	volatile	-0.6933	42	0.0	0.015	deepar	0
1	volatile	0.08	32	0.0	0.005	nhits	0
1	volatile	0.1051	8	0.9	0.042	tft	30
1	volatile	0.0636	44	0.0	0.001	nhits	0
1	volatile	0.0381	31	0.0	0.013	rnn	0
1	volatile	-0.8287	14	0.0	0.007	deepar	0
1	volatile	0.1191	3	0.3	0.037	tft	30
2	volatile	0.1097	23	0.0	0.0	nhits	0
2	volatile	0.0383	49	0.0	0.034	rnn	0
2	volatile	0.0827	3	0.3	0.037	tft	30
2	volatile	-1.0144	33	0.0	0.056	deepar	0

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
2	volatile	0.0165	36	0.0	0.066	rnn	0
2	volatile	0.0756	24	0.0	0.0	nhits	0
2	volatile	-0.6906	32	0.0	0.008	deepar	0
2	volatile	0.0642	2	0.0	0.001	tft	16
2	volatile	-0.6628	33	0.0	0.047	deepar	0
2	volatile	0.0621	4	0.2	0.001	tft	30
2	volatile	0.0172	42	0.0	0.052	rnn	0
2	volatile	0.1239	11	0.0	0.002	nhits	0
2	volatile	0.1074	27	0.0	0.001	nhits	0
2	volatile	-0.1538	42	0.0	0.047	deepar	0
2	volatile	0.0446	33	0.0	0.051	rnn	0
2	volatile	0.1103	2	0.0	0.001	tft	16
2	volatile	0.1217	0	0.3	0.082	tft	30
2	volatile	-0.4516	43	0.0	0.048	deepar	0
2	volatile	0.1313	45	0.0	0.001	nhits	0
2	volatile	0.0438	44	0.0	0.053	rnn	0
3	volatile	0.0965	43	0.0	0.0	nhits	0
3	volatile	0.081	4	0.2	0.001	tft	30
3	volatile	0.0502	26	0.0	0.015	rnn	0
3	volatile	-0.7591	23	0.0	0.015	deepar	0
3	volatile	-0.5881	13	0.0	0.006	deepar	0
3	volatile	0.019	41	0.0	0.026	rnn	0
3	volatile	0.0856	4	0.2	0.001	tft	30
3	volatile	0.0797	27	0.0	0.0	nhits	0
3	volatile	0.1611	24	0.0	0.0	nhits	0
3	volatile	0.0622	1	0.4	0.038	tft	30
3	volatile	0.0215	46	0.0	0.023	rnn	0
3	volatile	-1.1243	42	0.0	0.01	deepar	0
3	volatile	0.122	39	0.0	0.001	nhits	0
3	volatile	0.0285	45	0.0	0.042	rnn	0
3	volatile	-0.4661	30	0.0	0.014	deepar	0
3	volatile	0.0833	17	0.3	0.006	tft	30
3	volatile	0.0387	44	0.0	0.042	rnn	0
3	volatile	0.11	0	0.3	0.082	tft	30
3	volatile	0.1317	42	0.0	0.0	nhits	0
3	volatile	0.5901	47	0.0	0.013	deepar	0
5	volatile	0.163	24	0.0	0.002	nhits	0
5	volatile	0.0463	43	0.0	0.029	rnn	0
5	volatile	-0.2628	35	0.0	0.048	deepar	0
5	volatile	0.0767	0	0.3	0.082	tft	30
5	volatile	0.0746	31	0.0	0.0	nhits	0
5	volatile	-0.2776	29	0.0	0.021	deepar	0
5	volatile	0.0847	49	0.1	0.014	tft	30
5	volatile	0.0337	41	0.0	0.067	rnn	0

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
5	volatile	-0.3946	49	0.0	0.083	deepar	0
5	volatile	0.0798	3	0.3	0.037	tft	30
5	volatile	0.1749	41	0.0	0.0	nhits	0
5	volatile	0.0345	46	0.0	0.021	rnn	0
8	volatile	0.1838	47	0.0	0.0	nhits	0
8	volatile	0.097	4	0.2	0.001	tft	30
8	volatile	0.0532	45	0.0	0.091	rnn	0
8	volatile	-0.5068	31	0.0	0.045	deepar	0
8	volatile	0.0298	37	0.0	0.053	rnn	0
8	volatile	0.102	22	0.0	0.0	nhits	0
8	volatile	-0.0927	44	0.1	0.028	deepar	0
8	volatile	0.0987	24	0.0	0.008	tft	30
8	volatile	0.0892	34	0.3	0.02	tft	30
8	volatile	-0.2856	43	0.0	0.011	deepar	0
8	volatile	0.1929	41	0.0	0.001	nhits	0
8	volatile	0.0417	41	0.0	0.099	rnn	0

Table A.1: Hyperparameters after 40 individual trials across both volatile and non-volatile extrapolation periods with 4 different models and 43 different timeframes. The loss value for deepar is negative because it uses a negative binomial loss function.